DOI: 10.19650/j.cnki.cjsi.J2412714

基于拟水流算法在移动机器人路径规划中的应用*

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

摘 要:针对传统算法在移动机器人路径规划中存在搜索效率低的问题,提出了一种拟水流算法。该算法利用主流点搜索模型得到所有主流点。从起点逐步流动,通过拟水流避障算法和拟病毒算法进行避障,直至终点,并平滑处理所得路径。通过栅格法对多种地图环境进行建模,将拟水流算法与蚁群算法、Dijkstra 算法、Floyd 算法和 A* 算法的路径长度及运算时间进行对比仿真实验。实验结果显示,与获得最短路径和最少时间的 A* 算法相比,拟水流算法获得的平均路径长度减少了 2.40%~6.30%,平均用时减少了 35.71%~53.51%。最后,将拟水流算法应用于移动机器人 Turtlebot2,并与 A* 算法进行了对比实测实验。实验结果显示,拟水流算法相较 A* 算法,实测路径增加了 3.83%,寻路时间减少了 10.77%,拐点数减少了 42.86%。

关键词: 移动机器人;拟水流算法;路径规划;栅格法
中图分类号: TH166 TP242 **文献标识码**: A **国家标准学科分类代码**: 510.80

Research on path planning of mobile robot based on the stream algorithm

San Hongjun[1,2], Yang Xiaoyuan[2], Chen Jiupeng[1,2], Sun Haijie[2], Zhang Haobin[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: A stream algorithm is proposed to address the low search efficiency of traditional algorithms in mobile robot path planning. Firstly, the algorithm obtains all the main points through the main point search model. Flow step by step from the starting point, when a single obstacle is encountered, the stream obstacle avoidance algorithm is invoked to avoid the obstacle. When multiple obstacles are encountered, the pseudo-virus algorithm is called to mark these obstacles. Then, the stream obstacle avoidance algorithm is called to avoid obstacles until the end. Finally, the resulting path is smooth. A variety of map environments are modeled by using the grid method. The path length and running time of the stream algorithm are compared with those of the ant colony algorithm, Dijkstra algorithm, and Floyd algorithm in simulation studies. The testing results show that, compared with the A* algorithm which achieves the shortest path and the least time, the average path length obtained by the stream algorithm is reduced by 2.40%~6.30%, and the average time is reduced by 35.71%~53.51%. To test the application of the stream algorithm in the actual scene, it is applied to the mobile robot Turtlebot2 and conducted a comparative experiment with the A* algorithm. The experimental results show that, compared with the A* algorithm, the measured path is increased by 3.83%, the running time is reduced by 10.77%, and the number of inflection points is reduced by 42.86%.

Keywords: mobile robot; stream algorithm; path planning; grid method

0 引 言

导航作为移动机器人的核心,一直是机器人研究领域的核心技术之一,分为不基于地图的导航方式和基于地图的导航方式[1]。由于传统基于地图的方法对人力、物力和财力的需求巨大,且难以适应动态和大规模环境,研究人员开始探索基于深度学习、强化学习无地图路径规划方法,通过与环境的互动和自主探索来实现路径规划。提出了深度强化学习(deep reinforcement learning,

收稿日期:2024-04-09 Received Date: 2024-04-09
*基金项目:云南省基础研究计划项目(202301AU070059)、云南省科技厅重大专项项目(202002AC080001)资助

DRL)[2],DRL 是一种将深度学习(deep learning, DL)与强化学习(reinforcement learning, RL)相结合的算法[3],其中 DL 主要负责利用神经网络的感知功能对输入的未知环境状态提取特征,实现环境状态到状态动作值函数的拟合;而 RL 则负责根据深度神经网络的输出和一定的探索策略完成决策,从而实现状态到动作的映射,进而较好地满足机器人的移动需求[4]。但是这类方法在实际应用时存在学习时间长、探索能力差和奖励稀疏的问题[5]。

因此本文回归传统的导航方法,主流的自主导航方式以地图为基础,包括 4 个基本组成部分:1)感知[5],机器人利用其传感器提取有用信息;2)定位[6],机器人确定其在工作空间中的位置;3)认知和路径规划[7],机器人决定如何引导以实现其目标;4)运动控制[8],机器人调节其运动以完成所需的轨迹。根据环境信息的类型,其中,路径规划算法可分为两类: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)[9]。

在全局路径规划算法中,可以根据算法各自不同的特点划分为两类,分别是基于搜索和采样的路径规划算法、基于智能算法的路径规划算法[10]。其中基于智能算法的路径规划算法如蚁群算法 ACO[11]、粒子群算法 PSO[12]等;基于搜索和采样的路径规划算法有 Dijkstra 算法[13]、A* 算法[14]和 Floyd 算法[15]。

蚁群算法(ant colony optimization, ACO)是一种基于群体觅食的随机启发式搜索方法[16],具有较强鲁棒性和适应性。为了解决 ACO 中存在的局部优化、收敛性差、搜索效率低等问题[17-19],许多研究者一直致力于改进初始搜索策略、信息素更新规则、路径选择规则和死锁惩罚机制[20-22]。尽管上述文献已经改善了 ACO 的路径规划性能,但考虑最短路径长度因素时,仍存在安全因素无法保障的问题[23]。

Dijkstra 算法是一种典型的最短路径算法。针对传统 Dijkstra 算法路径规划速度慢、效率低[24]的问题,很多学者通过引入估计函数、路径平滑度,以时间权重为目标求解最优路径,这实现在缩短响应时间的基础上规划出最短路径[25]。

A* 算法[26]是一种基于几何的路径规划算法。针对在大场景中,其计算量大、计算时间长且效率不高等问题[27]。前人通过调整搜索邻域为无数个[28]、引入时间因子[29]等方法,A* 算法的规划策略与效率得以提高。

Floyd 算法是一种用于寻找给定加权路径拓扑网络中顶点间最短路径的算法[30]。针对路径规划时间复杂度较高、路径规划耗时长的问题[31],通过优化初始距离矩阵,剔除非合理路径点,减少判断次数来提高运算效率[32-34]。

移动机器人在路径规划时,首先通过感知模块检测环境并构建地图,定位自身位置。在静态环境下,先进行全局路径规划,以确保从起点到终点无碰撞、安全且最短。基于采样和搜索算法的简便性,在实际全局路径规划中广受欢迎。然而,这些全局路径规划算法也存在一些问题,如规划速度慢、效率低和路径不平滑。针对这些问题,本文提出了一种模拟水流流动来进行路径规划的拟水流算法,具体包含 1 个模型和 4 个子算法:1 种基于当前位置与终点的动态计算主流点的模型,通过探测、搜索、检测和转化 4 个步骤确定可行的主流点;1 种模拟病毒扩散传播过程的拟病毒算法,用于识别流动过程中的障碍物;1 种模拟水流流动轨迹避开障碍物的拟水流避障算法;2 种路径优化算法对已经完成规划的拟水流路径进行优化。通过与传统全局算法进行对比试验,表明拟水流算法能够有效地提高计算效率。

1 拟水流算法

1.1 基本原理

自然界中,水流在流淌过程中,能够巧妙地绕开乱石、树桩等障碍物,这种流动特性为移动机器人的路径规划过程提供了深刻的启示。通过观察水流的流动方式,本文将机器人的行走路径视作一条拟水流,模拟其从起点流向终点的过程。在这个过程中,障碍物如同阻碍水流前行的乱石,而机器人的路径规划则是在地图环境中模拟水流流动,寻找出一条安全可靠的通行路径。为方便理解,本文所提出算法将以移动机器人常用栅格地图为背景进行介绍。

拟水流算法由 1 个模型和 4 个子算法组成,在运用拟水流算法对移动机器人进行路径规划时,先调用主流点搜索模型搜索出所有主流点(包括起点和终点),且规定拟水流从起点出发,经过主流点,最终流向终点。随后拟水流开始从第 1 个主流点(起点)向最后 1 个主流点(终点)逐步流动,如果流动过程中遇到了障碍物,就调用拟水流避障算法进行避障,如果遇到了多个障碍物就先调用拟病毒算法标记相关障碍物,再调用拟水流避障算法通行,直至最后 1 个主流点。最后,再调用转折因子优化算法和平滑处理优化算法对初步完成规划的路径进行优化。

1.2 主流点搜索模型

把拟水流可以通行的点定义为主流点。通过寻找主流点,规定拟水流仅在多个主流点之间流动来限定拟水流总体流向。本文提出了一种基于当前位置与终点动态计算主流点的模型,具体计算过程如下所述:

如图 1 所示,先根据当前位置与终点的位置确定探测角度 θ_{d_k},$\theta_{d_k} \in [-90°, 90°]$,再根据探测角度确定搜

索角度 θ_{S_k},最后用搜索角度确定搜索步长 d_{S_k},以确定主流点。首先,用检测函数检测当前位置;其次,用转化函数将当前位置的坐标转化为该位置的序号并存储在主流点集合{M}中。在主流点集确定之后,把起点 S 添加到主流点集合中充当第 1 个主流点,把终点 E 添加到主流点集合中充当最后 1 个主流点。如果探测角度不在探测区间内,则采用镜像思维,从终点出发探测起点,最后将得到的路径进行倒置,这样就可以得到一条起点到终点的路径。在寻找主流点时参照主流搜索模型按式(1)~(8)进行更新计算。

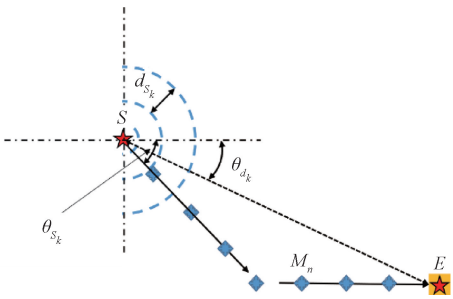


图 1　主流点搜索模型

Fig. 1　Mainstream points search model

$$D(P_n,P_{n-1})=\sqrt{(i_n-i_{n-1})^2+(j_n-j_{n-1})^2} \tag{1}$$

其中,(i_n,j_n) 是网格的位置坐标,n 表示当前位置的序号,$D(P_n,P_{n-1})$ 用来计算两个点之间的距离。

$$ind(i,j)_{transfer}=R\cdot(j_n-1)+i_n \tag{2}$$

其中,ind 用于计算网格的索引号,R 是行的数量,即通过上述获得的网格的索引号为基于网格从上往下从左往右的编号,例如对于 10×10 网格中坐标(5,2),其索引号为 15;

$$T(i_n)=\begin{cases}R, & \mathrm{mod}(ind,R)=0\\ \mathrm{mod}(ind,R), & 否则\end{cases} \tag{3}$$

$$T(j_n)=\begin{cases}\mathrm{int}\left(\dfrac{ind}{R}\right), & \mathrm{mod}(ind,R)=0\\ \mathrm{int}\left(\dfrac{ind}{R}\right)+1, & 否则\end{cases} \tag{4}$$

其中,横坐标 i_n 的转化函数为 $T(i_n)$,纵坐标 j_n 的转化函数为 $T(j_n)$,网格比例为 r,并根据机器人的尺寸决定;网格索引号为 ind;bmod( ) 为取余函数;int( ) 为取整函数;R 是总行数。

$$\theta_{d_k}=\arcsin\left(\frac{|j_{M_n}-j_E|}{D(P_k,E)}\right) \tag{5}$$

其中,θ_{d_k} 表示第 k 次迭代下的探测角度,(i_E,j_E) 是终点位置 E 的网格坐标,(i_k,j_k) 表示第 k 次迭代下当前位置 P_k 网格坐标。

$$\theta_{S_k}=\begin{cases}0^\circ, & \theta_{d_i}=0^\circ\\ 45^\circ, & \theta_{d_k}\in(0,90^\circ)\\ 90^\circ, & \theta_{d_k}=90^\circ\\ -45^\circ, & \theta_{d_k}\in(-90^\circ,0^\circ)\\ -90^\circ, & \theta_{d_k}=-90^\circ\end{cases} \tag{6}$$

$$d_{S_k}=\begin{cases}r, & \theta_{S_k}\in\{0^\circ,90^\circ/-90^\circ\}\\ \sqrt{2}r, & 其他\end{cases} \tag{7}$$

$$F_c(i_{k+1},j_{k+1})=\begin{cases}1, & feasible\\ 0, & 其他\end{cases} \tag{8}$$

其中,θ_{S_k} 表示第 k 次迭代下的搜索角度;d_{S_k} 是第 k 次迭代下的探测步长,基于第 k 次迭代下的当前位置按照搜索角度 θ_{S_k} 确定方向、以探测步长确定的搜索点作为第 $k+1$ 次迭代的当前位置(i_{k+1},j_{k+1});在本文中,设定网格比例 r 为 1,这意味着每个单元网格的边长与机器人的外接圆直径之间的比例是相等的。$F_c(i_{k+1},j_{k+1})$ 是搜索点检验函数,用来检验搜索点是否为主流点,即如果 (i_{k+1},j_{k+1}) 所代表的点为可通行区域,则 $F_c(i_{k+1},j_{k+1})$ 取值为 1,表明该搜索点为主流点;否则检测不通过。

寻找主流点过程如下:首先根据当前点与终点确定探测角度,再根据探测角度确定搜索角度,接着根据搜索角度确定搜索步长;其次,判断当前点是否满足主流点条件,若满足,则存入主流点集合中;最后判断是否已搜索到终点,如不是终点,则进入下一轮探测。

1.3　拟病毒算法

本文提出了一种模拟病毒扩散传播过程的拟病毒算法。该算法中把拟水流即将经过的第一个障碍物群定义为相关障碍物,把后续经过和不会经过的障碍物定义为无关障碍物。拟水流算法通过拟病毒算法来识别障碍物的轮廓。

拟病毒算法的实现:如图 2 所示,假设拟水流在流动时携带有一种病毒,当拟水流碰到第 1 个障碍物群时就会向距离拟水流最近障碍物栅格传播病毒,该病毒就会从携带病毒的栅格向周围(8 个方向)进行扩散感染,被感染的障碍物栅格又会向周围继续扩散感染,直到感染该区域内的全部障碍物。通过这种方式,拟水流就可以完整识别出当前障碍物的轮廓。

其中 d 是病毒扩散半径。为确保该区域内整个相关障碍物可以被病毒准确覆盖,而不会影响到该区群域内的其他障碍物,取 $d=\sqrt{2}r$,r 是网格的比例,由机器人的几何形状决定。病毒注入后会感染扩散范围内的障碍物,而扩散范围内的障碍物被感染后又会紧接着扩散感染,直到该相关障碍物被全部感染。

拟病毒算法过程如下:首先,将最接近机器人的一个障碍物视为病毒源;其次,以半径 d 向外扩散,若被覆盖

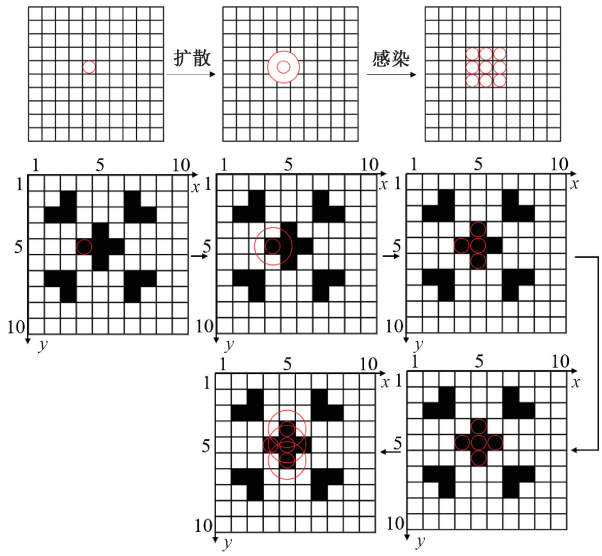


图 2 病毒扩散传播示意图

Fig. 2 Diagram of the virus algorithm

区域为相关障碍物,则感染这些相关障碍物,被感染的相关障碍物作为新的病毒源开始新一轮扩散,直至感染结束;若不是,则直接结束;最终得到被感染区域所有点的坐标。

图 3 展示了拟病毒算法仿真过程,向图示位置注入病毒,就会在该障碍物群内迅速扩散传播,经过数次迭代后,该障碍物群就会被拟病毒标记,从而被拟水流所识别。

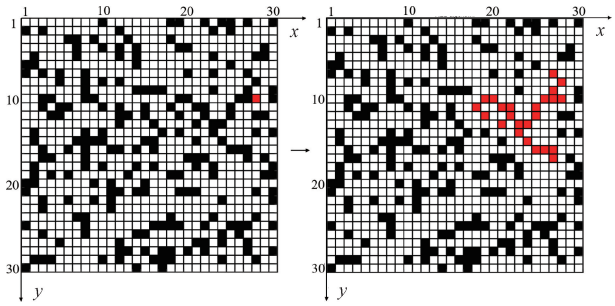


图 3 拟病毒算法效果仿真示意图

Fig. 3 Simulation diagram of the effect of the virus algorithm

1.4 拟水流避障算法

拟水流在遭遇障碍物时,需要巧妙地避开它们以继续其行进路线。为此,本文提出了一种新的避障方法,即拟水流避障算法。这种算法模拟了水流在自然环境中如何巧妙地绕开石头等障碍物的流动轨迹。拟水流避障策略如图 4 所示。在实施避障策略时,设定一个限制条件,如图 5 所示,在这里,障碍物被视为能够支撑拟水流流动的“石头”。当拟水流遇到“石头”时,它只能依附于障碍物的表面流动。这一设计不仅模拟了水流在现实中的流动特性,还确保了拟水流在避障过程中的连续性和稳定性。

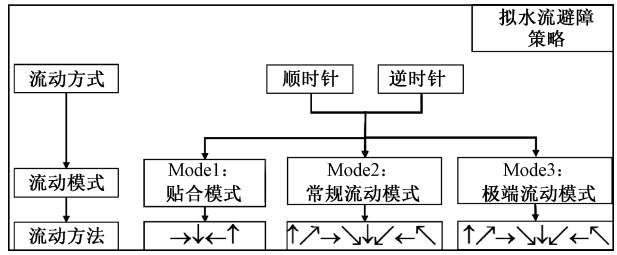


图 4 避障策略

Fig. 4 Obstacle avoidance strategy

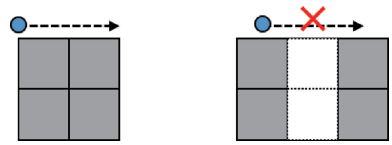


图 5 限定条件

Fig. 5 Limitation

在本文中规定了拟水流避障算法中的逻辑层次:流动方式、流动模式、流动方法。其中流动方法为最低一级,流动方式为最高一级。介绍如下:

拟水流在避障时有顺时针和逆时针两种基础流动方式,如图 6 所示。顺时针流动方式——由↑↗→↘↓↙←↖共 8 种流动方法组成;同理,逆时针流动方式——由↓↘→↗↑↖←↙共 8 种流动方法组成。

选定流动方式后,拟水流会按照该方式下的 3 种不同流动模式继续行进。每种流动模式的选择都需要满足特定的条件,这些条件确保了拟水流能够根据实际情况灵活调整其流动状态。

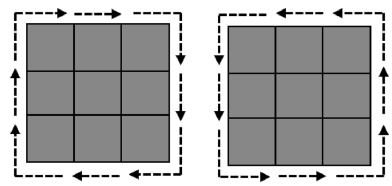


图 6 流动方式

Fig. 6 Types of flows

模式 1:若拟水流此刻处于障碍物的 4 个对角位置之一,如图 7、图 8(a)~(d)所示,拟水流将采取贴合障碍物的流动方式,本文称之为“贴合模式”。这样的流动策略有助于拟水流有效地避开障碍物,确保路径安全。

模式 2:其对应的流动方法和条件如图 7、8 中(e)~(l)所示,当拟水流接近障碍物时,它会开始绕过障碍物继续流动,本文称之为“常规流动模式”。

模式 3:拟水流即将离开栅格地图的那一刻所处的

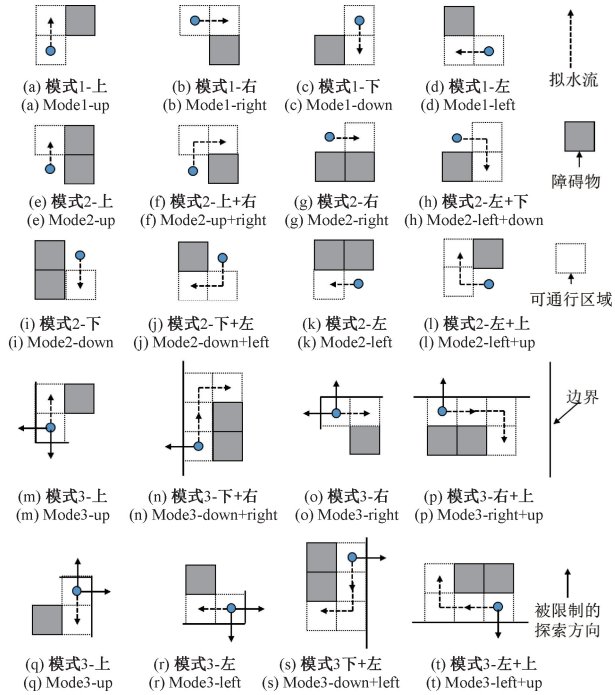


图 7　顺时针流动方式

Fig. 7　Clockwise flow mode

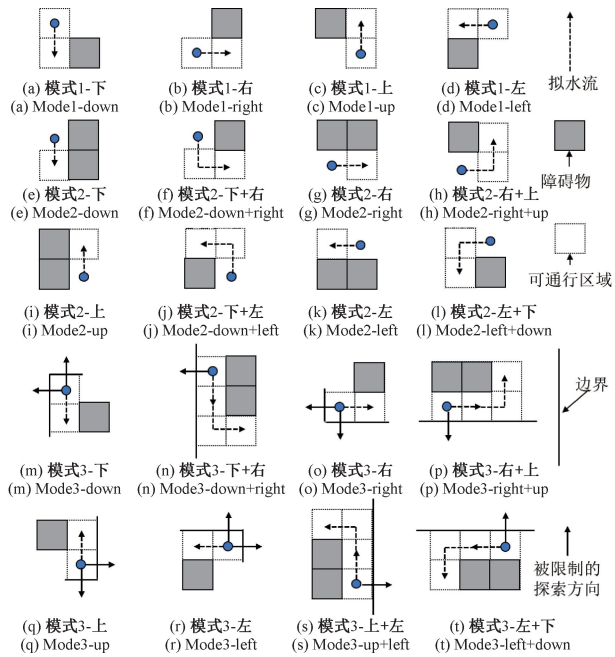


图 8　逆时针流动方式

Fig. 8　Counterclockwise flow mode

位置定义为“极端位置”。此时,如果继续采用常规模式将产生拟水流溢出地图区域、超出地图的索引值等结果。因此,在常规模式的基础之上增加了边界限定条件,调整了具体的流动条件,称之为“极端流动模式”。其对应的流动方法和条件如图 7、8 中(m)~(t)所示。不同流动条件对应的搜索方向及代码符号如表 1 所示(以向上搜索为北),表 2 和 3 列出了不同方式下各流动模式中的流动条件及方法对应的关系。

表 1　代码及方向关系

Table 1　Code and direction relationship

流动条件	搜索方向	符号
map(i, j+1)	向正东方向搜索	E
map(i+1, j)	向正南方向搜索	S
map(i, j−1)	向正西方向搜索	W
map(i−1, j)	向正北方向搜索	N
map(i+1, j+1)	向东南方向搜索	ES
map(i−1, j+1)	向东北方向搜索	EN
map(i+1, j−1)	向西南方向搜索	WS
map(i−1, j−1)	向西北方向搜索	WN
= 0	搜索结果为可通行区域	0
= 1	搜索结果为障碍物	1

表 2　顺时针中流动方法及条件

Table 2　Method and condition of clockwise flow

方法	模式	图例	边界限定条件	条件
↑	M1	Fig9(a)	\	N0, EN1, E0
	M2	Fig9(e)	\	E1, EN1, N0
	M3	Fig9(n)	j=0	E1, EN1, N0
	M3	Fig9(m)	j=0 & i=R	EN1, N0
↗	M2	Fig9(f)	\	E1, N0, EN0
	M3	Fig9(n)	j=0	E1, N0, EN0
→	M1	Fig9(b)	\	E0, ES1, S0
	M2	Fig9(g)	\	S1, ES1, E0
	M3	Fig9(p)	i=0	S1, ES1, E0
	M3	Fig9(o)	i=0 & j=0	ES1, E0
↘	M2	Fig9(h)	\	S1, ES0, E0
	M3	Fig9(p)	i=0	S1, E0, ES0
↓	M1	Fig9(c)	\	S0, WS1, W0
	M2	Fig9(i)	\	W1, WS1, S0
	M3	Fig9(s)	j=C	W1, WS1, S0
	M3	Fig9(q)	i=0 & j=C	WS1, S0
↙	M2	Fig9(j)	\	W1, S0, WS0
	M3	Fig9(s)	j=C	W1, WS0, S0
←	M1	Fig9(c)	\	W0, WN1, N0
	M2	Fig9(S)	\	N1, WN1, W0
	M3	Fig7(t)	i=R	N1, WN1, W0
	M3	Fig7(r)	i=R & j=C	WN1, W0
↖	M2	Fig7(l)	\	N1, W0, WN0
	M3	Fig7(t)	i=R	N1, WN0, W0

表 3 逆时针中流动方法及条件

Table 3 Method and condition of counterclockwise flow

方法	模式	图例	边界限定条件	条件
↓	M1	Fig10(a)	\	S0, ES1, E0
	M2	Fig10(e)	\	E1, ES1, S0
	M3	Fig10(n)	j=0	E1, ES1, S0
	M3	Fig10(m)	j=0 & i=0	ES1, S0
↘	M2	Fig10(f)	\	E1, ES0, S0
	M3	Fig10(n)	j=0	E1, S0, ES0
→	M1	Fig10(b)	\	E0, EN1, N0
	M2	Fig10(g)	\	N1, EN1, E0
	M3	Fig10(p)	i=R	N1, EN1, E0
	M3	Fig10(o)	i=R & j=0	EN1, E0
↗	M2	Fig10(h)	\	N1, E0, EN0
	M3	Fig10(p)	i=R	N1, E0, EN0
↑	M1	Fig10(c)	\	N0, WN1, W0
	M2	Fig10(i)	\	W1, WN1, N0
	M3	Fig10(s)	j=C	W1, WN1, N0
	M3	Fig10(q)	i=R & j=C	WN1, N0
↖	M2	Fig10(j)	\	W1, N0, WN0
	M3	Fig10(s)	j=C	W1, WN0, N0
←	M1	Fig10(c)	\	W0, WS1, S0
	M2	Fig10(d)	\	S1, WS1, W0
	M3	Fig10(t)	i=0	S1, WS1, W0
	M3	Fig10(r)	i=0 & j=C	WS1, W0
↙	M2	Fig10(l)	\	S1, W0, WS0
	M3	Fig10(t)	i=0	S1, WS0, W0

拟水流避障算法的运行流程为：

首先导入地图(包括子节点和目标点)。若子节点已临近目标点,则程序结束,否则执行流动模式 1,使子节点贴近障碍物,若子节点贴近障碍物,则执行流动模式 3,否则执行流动模式 2。随后更新子节点,并判断子节点是否临近目标点,开始下一轮循环。

以顺时针流动方式为例,详细解释探索过程如下:在流动模式 2 下,需要对 8 种不同的流动方法逐一进行条件判断的过程可以称之为“探索”。具体来说,系统会按照预设的顺序,逐一检验每一种流动方法所对应的流动条件是否成立,一旦找到符合某一流动方法对应的流动条件,系统将立即执行该流动方法。这种全面而细致的探索方式确保了流动模式的灵活性和准确性。而在流动模式 3 下,探索过程则有所不同。系统不再对每一种流动方法都进行条件判断,而是聚焦于满足特定边界条件的流动方法。只有当流动方法符合这些边界条件时,系统才会进行进一步探索。这种有针对性的探索方式提高了处理效率,使得流动过程更加高效和精准。不同的流动模式对应着不同的探索策略,旨在适应不同的应用场景和需求。

在执行拟水流避障算法时有两种路径保留情况：

情况 1:如图 9(a)和(d)所示,尝试从顺时针逆时针两个方向同时出发,如果都能得到可行的路径,那么只保留较短路径;

情况 2:如图 9(b)和(c)所示,当拟水流因边界或障碍物无法得到可行路径解时,算法将舍弃这一侧的路径,结果如如图 9(e)和(f)所示。

在规避障碍物的过程中,为确保路径的唯一性和连续性,设定了特定的路径点与集合命名规则。

每次搜索所得的路径点,称之为 $P(k)$。对于顺时针方向搜索到的路径点,将其集合命名为 *temp_path_left*;而对于逆时针方向搜索到的路径点,其集合则命名为 *temp_path_right*。在搜索过程中,若路径点满足 $P(k) \notin temp\_path\_left$,则将其添加至顺时针路径点集中,反之同理。这样的设定确保了搜索过程中的路径点不会重复,提高了避障的效率和准确性。

2 拟水流算法在路径规划中的应用

2.1 拟流水算法寻路过程

如图 10 所示,在主流点确定后,拟水流从第 1 个主流点出发,向第 2 个主流点流动;第 2 个主流点又向第 3 个主流点流动,直至流到最后 1 个主流点。在主流点之间流动时,拟水流会基于自己的位置探测下一个主流点 M_i 的位置,探测方向为指向 M_i 的方向。在探测过程中如果先找到主流点,则直接流向主流点,如果先找到障碍物再找到主流点,则需先判断拟水流是否受到无关障碍物影响,如果存在无关障碍物则调用拟病毒算法识别出障碍物的轮廓,然后再调用拟水流避障算法流向下一个主流点。如果不存在无关障碍物则直接调用拟水流避障算法流向下一个主流点。

如图 11,当拟水流遇到多个障碍物时会被分流或引发拟水流对冲而锁死,会影响搜索效率甚至会因此无法成功进行路径规划。通过将智能拟水流算法和拟病毒算法融合,拟水流就可以先完整识别出当前相关障碍物的轮廓,然后在没有无关障碍物干扰的情况下顺利高效地绕过障碍物。为了不影响下一次规划,在拟水流流过相关障碍物后,病毒会自动清除。

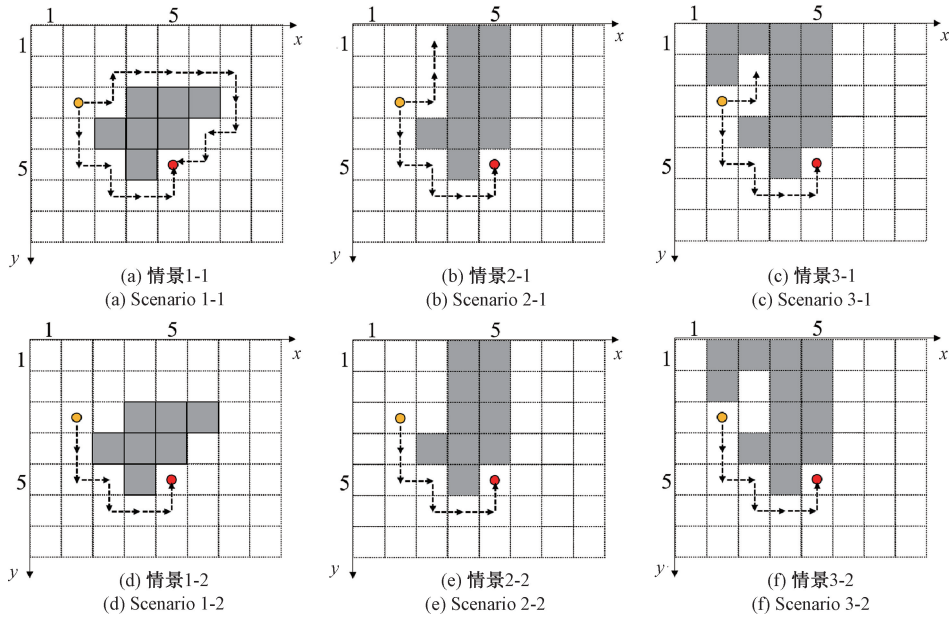


图 9　拟水流避障示意图

Fig. 9　Diagram of simulated water obstacle avoidance

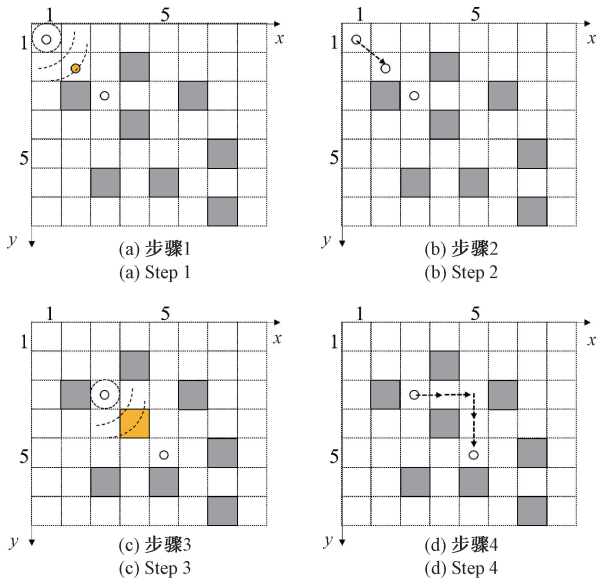


图 10　两种探测结果

Fig. 10　Two results of detection

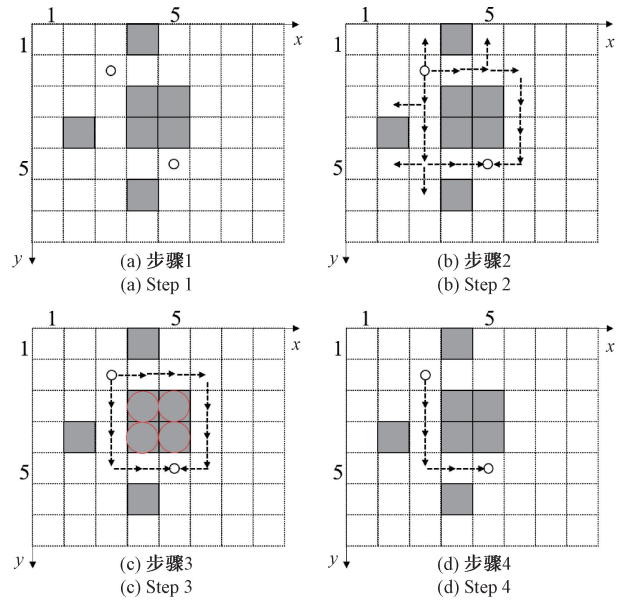


图 11　拟水流避障算法融合拟病毒算法示意图

Fig. 11　Diagram of the stream avoidance algorithm fused with the virus algorithm

2.2　优化拟水流路径

一方面,由于算法本身特性,所生成的路径往往包含大量转折点;另一方面,由于该算法基于障碍物进行路径规划,这导致生成的路径往往紧贴障碍物,这对于移动机器人的实际应用来说效果并不理想。鉴于此,特别提出了 2 种优化策略。在拟水流算法完成初步寻路后,对众多转折点进行优化处理,以改善路径的平滑性和提高机器人的移动效率。

1)转折因子优化算法:在路径集 $\{P\}$ 中找到不在同一条直线上的 3 个路径点 P_{k-1},P_k,P_{k+1}($k>2$)(此时线段 $P_{k-1}P_k$ 的斜率与线段 P_kP_{k+1} 不相等,即发生转折现象,P_k 被定义为转折因子)。判断是否可以从 P_{k-1} 到达 P_{k+1},如果可以则舍弃转折因子 P_k,依照此方法检测到终点,即可完成第一轮优化。具体优化过程如算法 1 所示。

2)平滑处理优化算法:首先,选定 $\{P\}$ 中的 P_1 作为平滑处理的起始点,P_2 作为第 2 个处理点;其次,P_1 往 P_2 方向扩展,同时判断扩展区内是否存在障碍物。若扩展

算法 1 转折因子优化算法

Algorithm 1 Transition factor optimization algorithm

算法 1：转折因子优化算法

Input：*Path*，*field*；
Output：*OptimalPath*；
Flag ← [ ]；*End* ← *numel*(*Path*)；*OptimalPath* ← *Path*
For k = 2 to *End* − 1
$(X_1,Y_1),(X_2,Y_2),(X_3,Y_3)$ ← *substitute Path*($k-1$), *Path*(k), *Path*($k+1$) into eq. 3,4
IF $(Y_1-Y_2)/(X_1-X_2) \neq (Y_2-Y_3)/(X_2-X_3)$
  For i = min(X_1,X_2) to max(X_1,X_2)
    For i = min(Y_1,Y_2) to max(Y_1,Y_2)
    If *field*(i,j) = 1
      Flag. *append*(1)
    break
    Else
      Flag. *append*(0)
    End if
  If 1 *in Flag*
    OptimalPath. *delete*(k)
    break
    End if
  End for
  End for
  End if
End for
Return *OptimalPath*

区内无障碍，则 P_1 往 P_k 方向继续扩展，否则，P_1 和 P_{k-1} 被纳入平滑路径集 $\{P_{smooth}\}$ 中。这一过程结束之后，开始下一次平滑处理，重复处理直至扩展到目标点结束，具体优化过程如算法 2、3 所示。

算法 2 平滑处理优化算法

Algorithm 2 Smoothing optimization algorithm

算法 2：平滑处理优化算法

Input：*OptimalPath*，*field*；
Output：*SmoothPath*；
NewPath ← [ ]，*Start* ← 1，*End* ← *numel*(*Path*)；
SmoothPath ← [ ]；*New_T* ← *OptimalPath*；
[*NewPath*，*d*] ← *FindNewPath*(*Start*, *End*, *field*, *New_T*)；
SmoothPath. *append*(*NewPath*)；
While (**True**)
  If d=*End*−1
    break
  End if
  [*NewPath*，*d*] ← *FindNewPath*(*Start*, *End*, *field*, *New_T*)；
  SmoothPath. *append*(*NewPath*)；
End while
SmoothPath. *append*(*NewPath*，*New_T*(*End*))；
Return *SmoothPath*

算法 3 *FindNewPath*

Algorithm 3 *FindNewPath*

算法 3：*FindNewPath*

Input：*field*，*Start*，*End*，*New_T*；
Output：*NewPath* ，*d*；
For d = *Start* + 1 *to End* − 1
  TestPoint_Start ← *New_T*(*Start*)；
  TestPoint_End ← *New_T*(d)；
  $(x,y),(X,Y)$ ← *substitute TestPoint_Start*，*TestPoint_End* into eq. 3,4；
  For i = min(x,X) to max(x,X)
    For j = min(y,Y) to max(y,Y)
    If *filed*(i,j) = 1
      Flag ← 1；
    break
    Else
      Flag ← 0
    End if
    If *Flag* = 1
      break
    End if
    End for
  End for
  break
  If *Flag* = 1
    NewPath. *append*(*TestPoint_Start*，*New_T*(d − 1))
  return
  End if
  If d = *End*
    NewPath. *append*(*TestPoint_Start*，*New_T*(d))
  return
  End if
End for
Return *NewPath*，*d*

2.3　适应特殊情况的算法改动

在理想条件下对栅格地图划分时,每一个栅格的尺寸一般是根据机器人实际尺寸划定的,要求机器人既能够占据一个栅格,同时又不会超出所占的栅格。但在实际应用中栅格地图的划定会给定一个静态地图分辨率(一般定义为 0.05 m),此时对于不同尺寸的机器人可能会占据多个栅格,在实际应用中一般采用加入障碍物膨胀半径使障碍物等比例向外膨胀的方法。但是,在障碍物比较集中的区域采用这个方法会使得碰撞区域覆盖本来可以通行的区域,使得机器人无路可走,此时就不会添加膨胀半径,但缺少了膨胀半径的地图中,如图 12 所示直接从障碍物的左侧斜向下到达目标点的做法是不可取的,会使的机器人与障碍物碰撞。为了应对障碍物相对集中时的情况,需要尽量避免使用倾斜方向的方法。此外,图 12 中,两个呈对角且相邻的障碍物视为阻塞状态,算法搜索的路径从这两个障碍物之间通过则视为失败。为满足不同情况的需要,本文针对拟水流算法将拟水流避障算法中流动方法的倾斜方向拆分为两个方向上的组合,拆分后的流动方法对应的流动条件与拆分前的一致。具体拆分为如表 4 所示。并将应对密集障碍物时有拆分情况的拟水流算法称为 SSA,而不需要拆分的拟水流算法称为 SA。

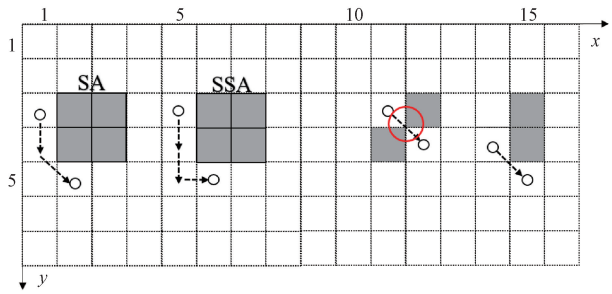


图 12　拟水流算法的两种情况

Fig. 12　Two cases of the stream algorithm

表 4　适应特殊情况的算法改动

Table 4　Algorithm changes for special cases

顺时针拆分		逆时针拆分	
↗	↑+→	↗	→+↑
↘	→+↓	↘	↓+→
↙	↓+→	↙	→+↓
↖	←+↑	↖	↑+←

注:"+":代表了先后顺序,例如↑+→表示先↑后→。

在经过拟水流优化后,得到最终路径。拟水流算法流程图如图 13 所示。

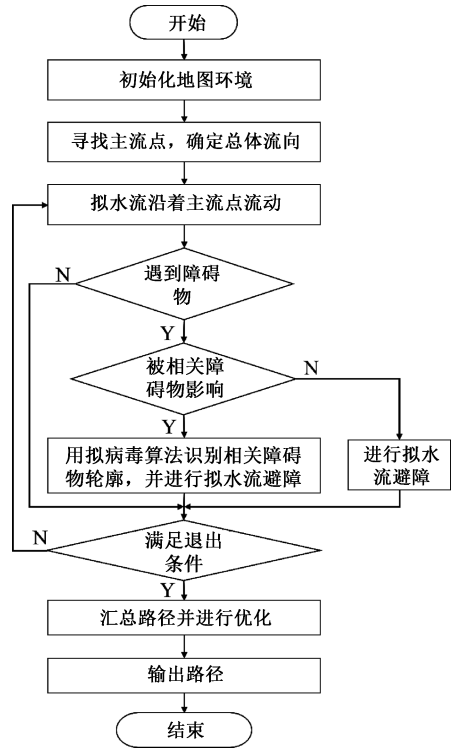


图 13　拟水流算法流程图

Fig. 13　Flowchart of the stream algorithm

3　仿真分析及实验验证

3.1　仿真环境地图建模

本文采用栅格法构建环境地图,此法因简洁高效、对障碍物适应性强,备受认可,在路径规划中常被视作优选方案[35]。图 14 所示的 10×10 的网格即可作为一个机器人的工作栅格地图,其中黑色区域代表障碍物,用逻辑值 1 标识,而白色区域则代表机器人可安全通过的可行域,并以逻辑值 0 作为标识[36]。

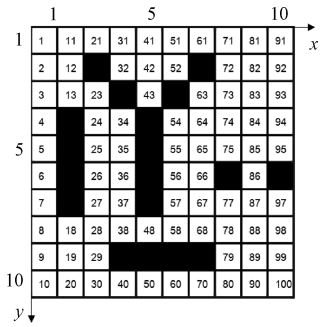


图 14　仿真环境

Fig. 14　Simulation environment

每个网格均被赋予了唯一且特定的序列号与坐标,确保了其空间位置的精准识别。网格序列号的排序严格

遵循从上至下、从左至右的顺序,使得每一个网格都能根据其位置获得一个明确的标识。网格序号与坐标之间的转换关系可见式(2)~(4)。

3.2 多类地图路径规划性能比较

考虑到本文提出的算法与对比的算法之间的算法检验标准相同点较少,在比较时将着重对路径长度、是否能够找到可行解、是否会发生碰撞和搜索时间 4 个衡量指标进行分析比较。为确保运算结果的可靠性,在同一张栅格地图上针对每个算法都重复执行 100 次,并对着 100 次运算结果进行平均处理。为了验证算法的通用性,本文设置了 2 组分别进行测试:1)障碍物密集且复杂的小型地图(包括迷宫类图形、20×20 复杂图形);2)障碍物相对稀疏的大型地图(50×50、100×100、150×150 和 200×200 网格地图)。

为满足不同情况的需要,将应对密集障碍物时有拆分情况的拟水流算法称为 SSA,而不需要拆分的拟水流算法称为 SA。

1)迷宫类地图:图 15 所展示的迷宫类地图属于移动机器人寻路问题中得到可行解比较困难的一种地图类型。针对此类地图的寻路过程,算法计算时间更长,而且对算法的综合性能提出了更高的要求,表 5 列出了 6 种算法在这类地图上的具体运算时间和所规划路径的长度。

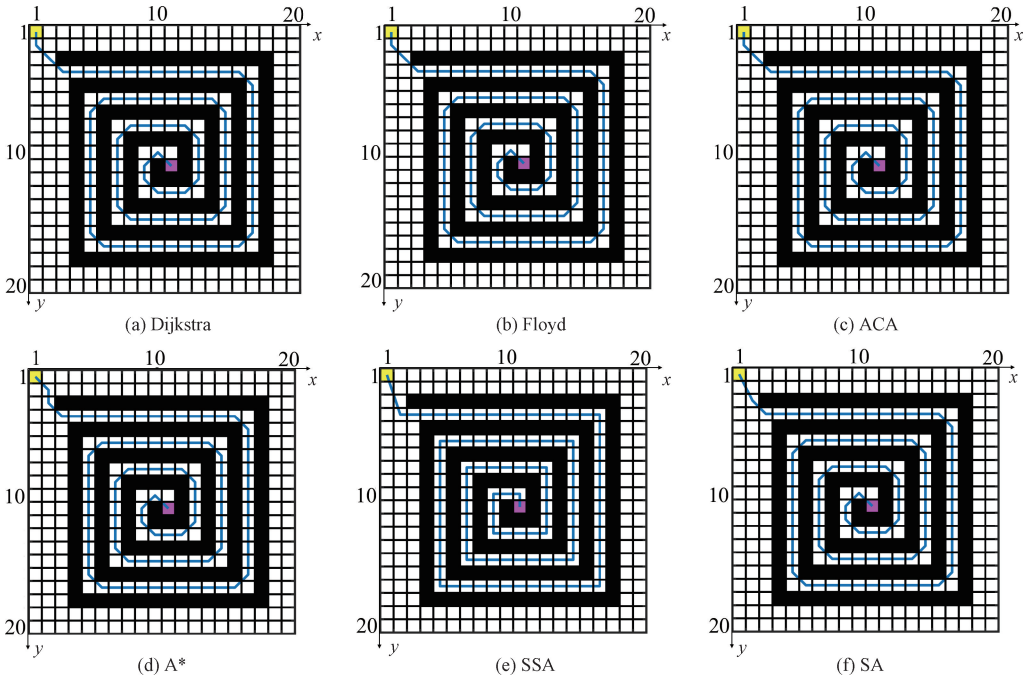


图 15 迷宫类地图环境

Fig. 15 The maze map

表 5 算法性能对比分析(迷宫类地图)

Table 5 Comparative analysis of algorithm performance (maze map)

算法	路径长度(cell)	寻路失败	碰撞	搜索时间/s
Floyd	101.21	N	N	0.73
Dijkstra	101.21	N	N	0.20
ACA	101.21	N	N	7.38
A*	101.21	N	N	0.11
SSA	108.41	N	Y	0.12
SA	101.03	N	N	0.10

由上述仿真实验可知,拟水流算法(SSA&SA)在复杂地形中的适应性较好,能够在较短的时间内得出路径长度较短的路径解,且针对 SSA 和 SA,只要 SA 能够得到可行解,SSA 也一定能得到可行解。

2)20×20 复杂环境地图:图 16 展示了随机生成的 20×20 复杂地图中,这 6 种算法的寻路结果,表 6 给出了这 6 种算法的路径规划长度、路径规划结果、运算时间。据 2.3 节所述,算法 ACA 与算法 A* 规划的路径通过圆圈所示点时,判定为寻路失败。

如图 16 所示,只有 Floyd 算法、Dijkstra 算法、SSA 算法和 SA 算法得到了可行解,从路径长度上分析,Floyd 算法、Dijkstra 算法得到的路径长度最短,都是 31.00;SA 算法得到的路径长度为 31.30,多了 0.96%;而 SSA 算法得到的路径长度最长,为 36.53,但是 SSA 算法得到的是一条安全无碰撞的路径。从时间上分析,Floyd 算法平均用时为 0.72 s,Dijkstra 算法平均用时为 0.13 s,是 SSA 算法

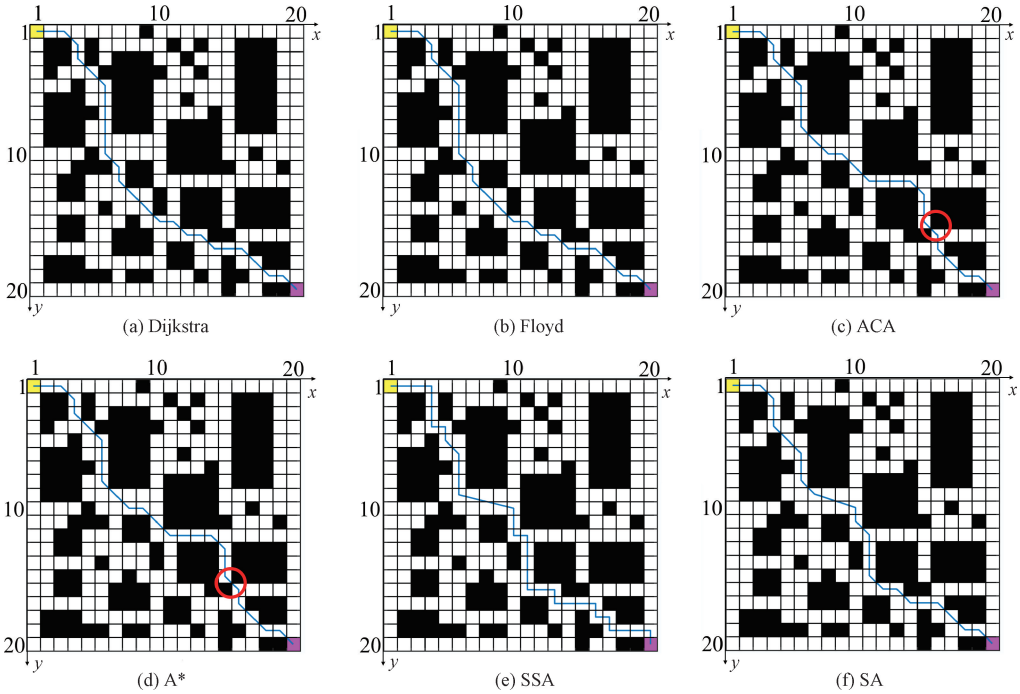


图 16　20×20 复杂地图环境

Fig. 16　20×20 complex map

表 6　算法性能对比分析(20×20 复杂地图)

Table 6　Comparative analysis of algorithm performance (20×20 complex map)

算法	Floyd	Dijkstra	ACA	A*	SSA	SA
路径长度/(cell)	31.00	31.00	\	\	36.53	31.30
寻路失败	N	N	Y	Y	N	N
无碰撞	N	N	N	N	Y	N
搜索时间/s	0.72	0.13	\	\	0.01	0.01

和 SA 算法的 2.35 倍。由此可知,在 20×20 复杂地图中,与 Floyd 算法、Dijkstra 算法相比,拟水流算法花费了最少的时间得出了相对较优的可行解。

3.3　综合比较分析

为了验证在机器人路径规划问题上,本文算法的可靠性和有效性,本文进行了多次模拟实验,以在不同的工作空间(50×50、100×100、150×150 和 200×200 网格图)中分别对拟水流算法和 A* 算法进行了 100 次仿真模拟,为了进一步保证算法数据对比的公平性,采用箱线法进一步分析两种算法运行时间的波动情况,以得出两种算法的性能差异。仿真结果如图 17 所示。

结合表 7、8 和图 18 可知,在 50×50 地图中,与 A* 算法相比,SA 算法的计算时间减少了 35.71%,路径长度数减少了 2.42%;在 100×100 地图中,SA 算法的计算时间减少了 52.55%,路径长度减少了 4.16%;在 150×150 地图中,SA 算法的计算时间减少了 4.25%,路径长度减少了 3.91%;在 200×200 地图中,SA 算法的计算时间减少了 53.51%,路径长度减少了 6.30%;由此可知,随着地图尺寸增加,相较于 A* 算法,拟水流算法的计算效率有着更高的提升。从而得出,相比于 A* 算法,SA 算法在大型地图中可以得到一条搜索时间更少且长度更短的路径,而且在复杂地形中的适应性也较强,更适用于实际的移动机器人路径规划。

3.4　实验验证

实验中,采用了 A2 激光雷达进行定位与感知,外接搭载了 Ubuntu18.04,melodic 版本 ROS 的笔记本电脑作为计算平台,负责运行机器人控制程序与感知算法。通过两根数据线,激光雷达与移动机器人底盘 TurtleBot2 得以与笔记本电脑紧密连接,共同构建了一个高效且稳定的实验平台。在实验开始之前先利用 amcl 和 gmapping 模块进行定位和建图,后通过注册 plugin 插件将拟水流算法植入 move_base 模块中,以便于替换全局路径规划器插件 global_planner,从而实现本文算法对实际环境的全局寻路过程。导航框架如图 19 所示,在实验开始之前,对 Turtlebot2 的建图功能进行了全面的测试。

测试地点选在实验室外的空旷楼道,以确保测试环境具有代表性。测试结果显示,Turtlebot2 能够准确地构建出楼道环境的地图,为后续的全局路径规划提供了坚实的基础。测试结果见图 20。

如图 21 所示,在走廊中放置了 5 个障碍物作为实验

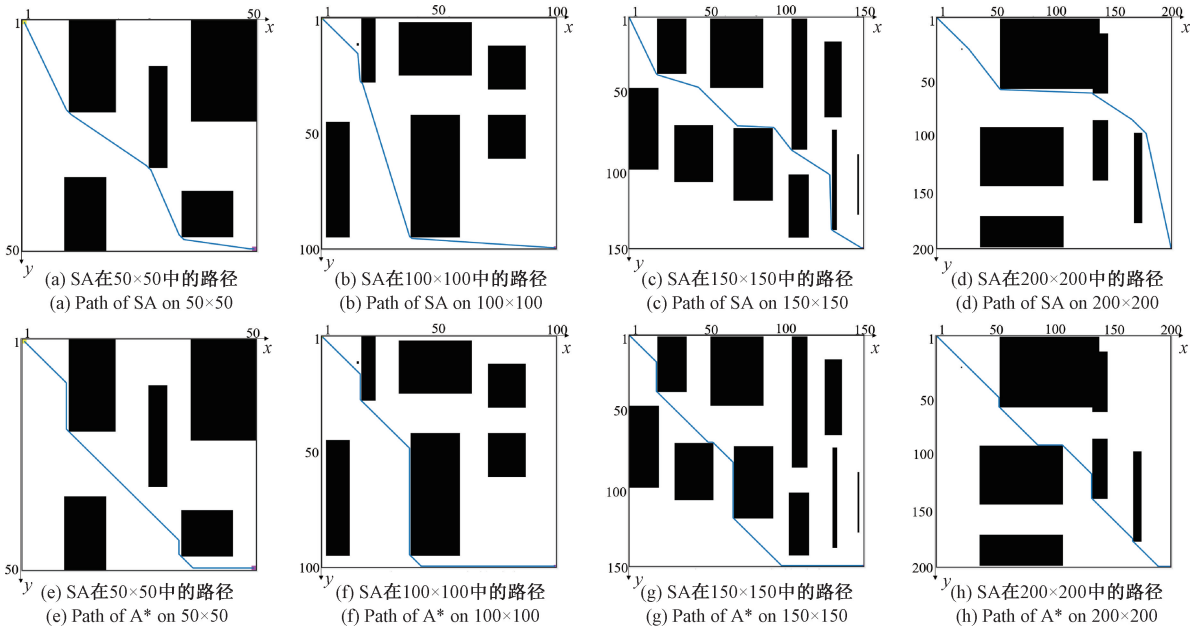


图 17 不同环境中 SA 与 A* 路径规划

Fig. 17 Path planning of SA and A* in different environments

表 7 算法时间对比(A* 与 SA)

Table 7 Comparison of algorithm time (A* and SA)

算法	地图尺寸	平均值	STD	MAX	MIN
A*/ms	50×50	12.60	0.89	14.6	11.9
	100×100	37.30	1.10	39.90	35.60
	150×150	72.90	4.00	77.80	70.60
	200×200	119.50	2.40	128.30	115.30
SA/ms	50×50	8.10	0.42	14.0	7.30
	100×100	17.70	0.97	23.80	16.70
	150×150	69.80	1.50	90.10	65.10
	200×200	55.70	1.70	61.30	53.30

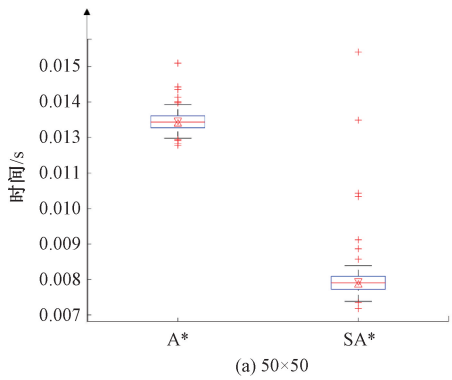
表 8 路径长度对比(A* 与 SA)

Table 8 Comparison of path length (A* and SA)

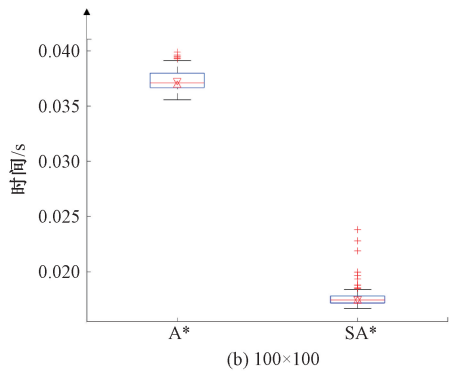
地图尺寸	A*(cell)	SA(cell)
50×50	76.91	75.05
100×100	173.39	166.18
150×150	242.93	233.43
200×200	319.72	299.58

场景。首先,通过键盘控制 Turtlebot2 移动,并利用机器人搭载的 A2 激光雷达扫描实验环境,构建该实验场景二维地图;其次,利用 map_server 模块保存所得地图;接着,通过导航节点程序导入地图,并采用 2D_Nav_Estimate 功能校准 Turtlebot2 的位置。最后,通过 move_base 模块中的拟水流算法在所得地图中进行路径规划,如图 22 所示,下侧点为起点,上侧点为目标点。

在实际应用中,A* 算法因其出色的性能而被广泛应用。为了全面评估本文所提的拟水流算法,将其与 A* 算法进行了对比实验,图 22(a)、(b)分别为拟水流算法路



(a) 50×50



(b) 100×100

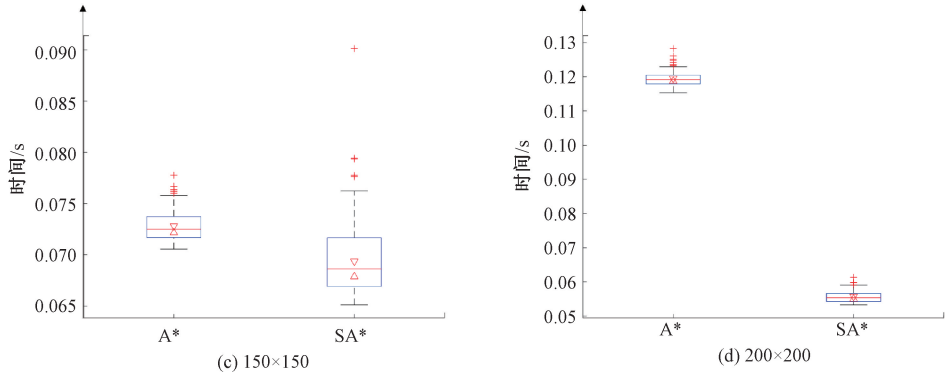


图 18　不同地图中算法时间对比(A* 与 SA)

Fig. 18　Comparison of algorithm time in different environments (A* and SA)

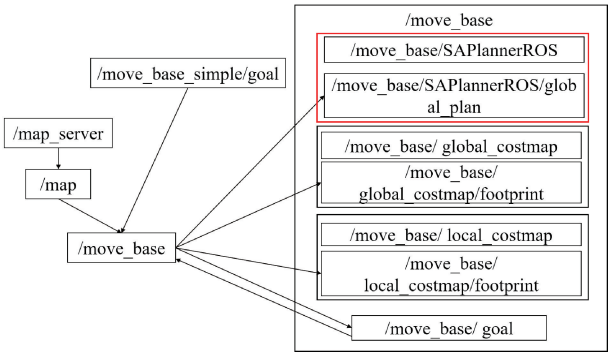


图 19　全局路径规划器插件框架

Fig. 19　Global path planner plug-in framework

图 21　实验场景

Fig. 21　Field experiment

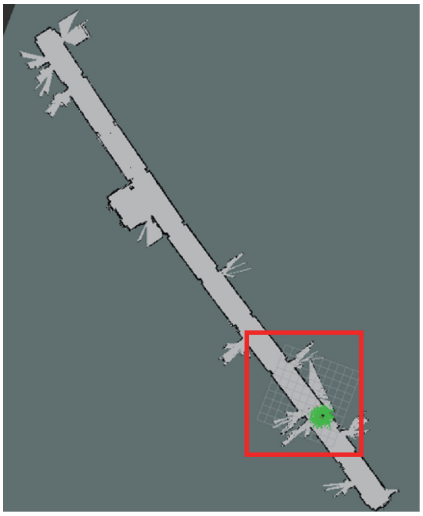
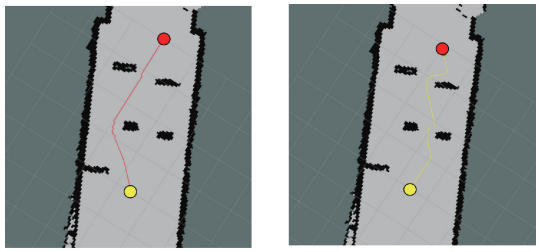


图 20　建图测试

Fig. 20　Test for mapping



(a) 拟水流算法　(b) A*

(a) SA　(b) A*

图 22　拟水流算法的 Turtlebot2 路径规划结果

Fig. 22　Path planning result of the stream algorithm on Turtlebot2

径和 A* 算法路径,表 9 为 2 种算法针对寻路时间、路径长度、路径转折点数的结果对比。实验结果显示,SA 算法相较 A* 算法,实测路径增加了 3.83%,但寻路时间减

表 9　实测算法性能对比(A* 与 SA)

Table 9　Performance comparison of measured algorithms (A* and SA)

算法	时间/ms	实测路径长度/mm	拐点数
SA	26.5	4 918.37	12
A*	29.7	4 736.84	21

少了 10.77%,拐点数减少了 42.86%。因 SA 算法选择的主流点偏向中间障碍物的左侧,导致该路径并不是最优路径,造成了实测路径长度和寻路时间相较 A* 算法,提升并不是很明显。由于本文算法引入了转折因子优化算法和路径平滑处理算法,导致在此次对比实验中,路径转折点数减小明显。总体而言,SA 算法所得路径相较于 A* 算法,具有更少的转折点和更短的寻路时间,并能寻路成功。这充分证明了 SA 算法在机器人寻路任务中的有效性与可行性。

4 结 论

针对传统算法在全局静态路径规划中的一些不足,本文提出了一种拟水流算法。通过模拟水流流动过程来进行路径规划,该算法可以在较短的时间内得到一条安全无碰撞的路径。首先按照主流点搜索模型,通过探测、搜索、检测和转化来确定出可行的主流点,再让水流沿着主流点流动,直到终点。为了解决拟水流在流过障碍物时容易受到无关障碍物的影响后分流和锁死的问题,将拟水流避障算法和拟病毒算法融合,使拟水流能够准确识别相关障碍物后迅速通行。最后,为了减少拟水流轨迹中的冗余路径对拟水流轨迹进行了优化。通过与 ACA 算法、Dijkstra 算法、A* 算法和 Floyd 算法的仿真实验对比分析,在保证路径解的质量的同时,拟水流算法的搜索速度有明显的提升,满足路径优化、高效计算和运行稳定等多重性能要求。在应对复杂的机器人工作环境或要求较高的寻路时效性的应用场景中,拟水流算法展现出了显著的应用前景。虽然本文的算法主要解决了静态环境下的全局路径规划问题,但其在动态环境或高维地图中的适应能力仍然存在继续研究的空间。

参考文献

[1] YAN CH L, CHEN G ZH, LI Y, et al. Immune deep reinforcement learning-based path planning for mobile robot in unknown environment [J]. Applied Soft Computing, 2023, 145: 110601.

[2] BELLMAN R E. A markovian decision process [J]. Indiana University Mathematics Journal, 1957, 6: 679-684.

[3] KROESE B J A. Learning from delayed rewards [J]. Robotics & Autonomous Systems, 1995, 15(4): 233-235.

[4] VOLODYMYR M, KORAY, K, DAVID S, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-534.

[5] 邓修朋,崔建明,李敏,等. 深度强化学习在机器人路径规划中的应用[J]. 电子测量技术,2023,46(6): 1-8.
DENG X P, CUI J M, LI M, et al. Application of deep reinforcement learning in robot path planning [J]. Electronic Measurement Technology, 2019, 46(6): 1-8.

[6] LIU L X, WANG X, YANG X, et al. Path planning techniques for mobile robots: Review and prospect [J]. Expert Systems with Applications, 2023, 227: 120254.

[7] PATLE B K, BABU G L, PANDEY A, et al. A review: On path planning strategies for navigation of mobile robot[J]. Defence Technology, 2019, 15(4): 582-606.

[8] CHRISTENSEN L, FERNANDEZ J D G, HILDEBRANDT M, et al. Recent advances in AI for navigation and control of underwater robots[J]. Current Robotics Reports, 2022, 3(4): 165-175.

[9] 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 H X, XIANG D, OUYANG J, et al. A review of research on path planning algorithms for mobile robots[J]. Computer Engineering and Applications, 2021, 57(18): 38-48.

[10] MAC T T, COPOT C, TRAN D T, et al. Heuristic approaches in robot path planning: A survey [J]. Robotics and Autonomous Systems, 2016, 86: 13-28.

[11] DORIGO M, GAMBARDELLA L M. Ant colony system: A cooperative learning approach to the traveling salesman problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53-66.

[12] SONG B Y, WANG Z D, ZOU L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve [J]. Applied Soft Computing, 2021, 100: 106960-106970.

[13] BAEZA D, IHLE C F, ORTIZ J M. A comparison between ACO and Dijkstra algorithms for optimal ore concentrate pipeline routing [J]. Journal of Cleaner Production, 2017, 144(15):149-160.

[14] 孙海杰,伞红军,肖乐. RRT-QSA*:一种改进的移动机器人路径规划算法[J/OL]. 系统仿真学报,1-17 [2023-08-22].
SUN H J, SAN H J, XIAO L. RRT-QSA*: An improved path planning algorithm for mobile robots[J/OL]. Journal of System Simulation, 1-17[2023-08-22].

[15] 郝自军,何尚录. 最短路问题的 Floyd 算法的若干讨论[J]. 重庆工学院学报(自然科学版),2008(5): 156-159.
HAO Z J, HE SH L. Some discussions of Floyd′s algorithm for the shortest-circuit problem[J]. Journal of Chongqing Engineering College (Natural Science

Edition), 2008(5):156-159.

[16] 孙凌宇,王威,秦红亮,等. 跳点优化蚁群算法的移动机器人路径规划[J]. 电子测量技术,2023,46(9):48-53.
SUN L Y, WANG W, QIN H L, et al. Path planning of mobile robot based on hopping point optimization ant colony algorithm [J]. Electronic Measurement Technology, 2019, 46(9): 48-53.

[17] 刘礼,刘勇,孙云权,等. 基于自适应蚁群算法的 AGV 路径规划优化[J]. 电子测量技术,2023,46(18):100-107.
LIU L, LIU Y, SUN Y Q, et al. AGV path planning optimization based on adaptive ant colony algorithm[J]. Electronic Measurement Technology, 2019, 46(18): 100-107.

[18] 魏立新,张钰锟,孙浩,等. 基于改进蚁群和 DWA 算法的机器人动态路径规划[J]. 控制与决策,2022,37(9):2211-2216.
WEI L X, ZHANG Y K, SUN H, et al. DWA algorithm based on improved ant colony optimization of robot dynamic path planning[J]. Control and Decision, 2022, 37(9): 2211-2216.

[19] 肖金壮,余雪乐,周刚,等. 一种面向室内 AGV 路径规划的改进蚁群算法[J]. 仪器仪表学报,2022,43(3):277-285.
XIAO J ZH, YU X L, ZHOU G, et al. An improved ant colony algorithm for indoor AGV path planning [J]. Chinese Journal of Scientific Instrument, 2022, 43(3): 277-285.

[20] HIDALGO-PANIAGVA A, VEGA-RODRIGUE M A, FERRUZ J, et al. Mosfla-mrpp: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning [J]. Engineering Applications of Artificial Intelligence, 2015, 44: 123-136.

[21] 江明,王飞,葛愿. 基于改进蚁群算法的移动机器人路径规划研究[J]. 仪器仪表学报,2019,40(2):113-121.
JIANG M, WANG F, GE Y, et al. Research on path planning of mobile robot based on improved ant colony algorithm[J]. Chinese Journal of Scientific Instrument, 2019,40(2):113-121.

[22] KURDI M. Ant colony system with a novel non-daemonactions procedure for multiprocessor task scheduling in multistage hybrid flow shop[J]. Swarm and Evolutionary Computation, 2019, 44: 987-1002.

[23] 申铉京,施英杰,黄永平,等. 基于双向蚁群算法的路径规划研究[J]. 哈尔滨工程大学学报,2023,44(5):865-875.
SHEN X J, SHI Y J, HUANG Y P, et al. Path planning based on bidirectional ant colony algorithm[J]. Journal of Harbin Engineering University, 2023, 44(5): 865-875.

[24] ZHOU Y L, HUANG N N. Airport AGV path optimization model based on ant colony algorithm to optimize Dijkstra algorithm in urban systems [J]. Sustainable Computing: Informatics and Systems, 2022, 35:100716.

[25] 罗强,王海宝,崔小劲,等. 动态环境下改进人工势场法的仓储机器人自主导航系统研究[J]. 计算机应用研究,2020,37(3):745-749,762.
LUO Q, WANG H B, CUI X J, et al. Research on autonomous navigation system of storage robot based on improved artificial potential field method in dynamic environment[J]. Application Research of Computers, 2020,37(3):745-749,762.

[26] 迟旭,李花,费继友. 基于改进 A~ * 算法与动态窗口法融合的机器人随机避障方法研究[J]. 仪器仪表学报,2021,42(3):132-140.
CHI X, LI H, FEI J Y. Research on robot random obstacle avoidance method based on the fusion of improved A ~ * algorithm and dynamic window method[J]. Chinese Journal of Scientific Instrument, 2021, 42(3): 132-140.

[27] 孙岩霆,王荣杰,蒋德松. 融合 A~ * 与 DWA 算法的水面船艇动态路径规划[J]. 仪器仪表学报,2024,45(1):301-310.
SUN Y T, WANG R J, JIANG D S. Dynamic path planning of surface vessel based on A ~ * and DWA algorithm[J]. Chinese Journal of Scientific Instrument, 2024, 45(1): 301-310.

[28] 辛煜,梁华为,杜明博. 一种可搜索无限个邻域的改进 A* 算法[J]. 机器人,2014,36(5):627-633.
XIN Y, LIANG H W, DU M B. An improved A* algorithm for searching infinite neighborhoods [J]. Robot, 2014,36(5):627-633.

[29] 张新艳,邹亚圣. 基于改进 A* 算法的自动导引车无碰撞路径规划[J]. 系统工程理论与实践,2021,41(1):240-246.
ZHANG X Y, ZOU Y SH. Collision-free path planning for automated guided vehicles based on improved A* algorithm[J]. Systems Engineering Theory and Practice, 2021,41(1):240-246.

[30] 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 R, ZHANG W SH. Smooth path planning of mobile robot based on improved ant colony algorithm [J].

Journal of Graphics, 2019,40(2):344-350.
[31] 石为人,王楷. 基于 Floyd 算法的移动机器人最短路径规划研究[J]. 仪器仪表学报,2009,30(10):2088-2092.
SHI W R, WANG K. Floyd′s algorithm-based shortest path planning for mobile robots[J]. Chinese Journal of Scientific Instrument, 2009,30(10):2088-2092.
[32] 王承平. 论移动机器人的智能路径规划算法综述[J]. 时代汽车,2022(6):13-14.
WANG CH P. A review of intelligent path planning algorithms for mobile robots[J]. Auto Time, 2022(6): 13-14.
[33] 王靖东. 基于优化 Floyd 算法的室内机器人路径规划研究[D]. 杨凌:西北农林科技大学, 2015.
WANG J D. Research on indoor robot path planning based on optimized Floyd algorithm [D]. Yangling: Northwest Agriculture and Forestry University, 2015.
[34] WANG J K, CHI W ZH, LI CH M, et al. Neuralrrt*: Learning-based optimal path planning[J]. Transactions on Automation Science and Engineering, 2020, 99: 1-11.
[35] LI Y J, WEI W, GAO Y, et al. PQ-RRT*: An improved path planning algorithm for mobile robots[J]. Expert Systems with Application, 2020, 152: 113425.
[36] ZHANG Y, WANG F L, FU F K, et al. Multi-agv path planning for indoor factory by using prioritized planning and improved ant algorithm[J]. Journal of Engineering and Technological Sciences, 2018, 50(4): 534-547.

作者简介

伞红军,2000 年于东北农业大学获得学士学位,2003 年于哈尔滨工业大学获得硕士学位,2009 年于哈尔滨工业大学获得博士学位,现为昆明理工大学机电工程学院副教授,主要研究方向为机器人技术理论及应用研究、机电产品设计与开发、现代农业装备研发等。
E-mail: sanhongjun@163.com

San Hongjun received his B. Sc. degree from Northeast Agricultural University in 2000, received his M. Sc. degree from Harbin Institute of Technology in 2003, and received his Ph. D. degree from Harbin Institute of Technology in 2009. He is currently an associate professor in Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology. His main research interests include robot technology theory and application research, mechanical and electrical product design and development, modern agricultural equipment research and development.

陈久朋(通信作者),2016 年于昆明理工大学获得学士学位,2018 年于昆明理工大学获得硕士学位,2021 年于昆明理工大学获得博士学位,现为昆明理工大学机电工程学院讲师,主要研究方向为机器人技术及应用、机电一体化产品开发。
E-mail: 18314490225@163.com

Chen Jiupeng (Corresponding author) received his B. Sc. degree from Kunming University of Science and Technology in 2016, received his M. Sc. degree from Kunming University of Science and Technology in 2018, and received his Ph. D. degree from Kunming University of Science and Technology in 2021. He is currently a lecturer in Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology. His main research interests include robot technology and application, mechatronics product development.

孙海杰,2020 年于浙江理工大学获得学士学位,现为昆明理工大学机械工程专业硕士研究生,主要研究方向为机器人导航、路径规划。
E-mail: haijiesun402@163.com

Sun Haijie received his B. Sc. degree from Zhejiang Sci-Tech University in 2020. He is currently a master student in mechanical engineering at Kunming University of Science and Technology. His main research interests include robot navigation and path planning.