DOI:10.13878/j.cnki.jnuist.20230218001

付雄[1]　李涛[1,2]

基于自适应步长策略的 A* 算法路径规划优化

摘要

针对 A* 算法求解路径轨迹耗时长、内存占用大等问题,本文提出一种基于自适应步长策略改进 A* 算法.首先,根据当前点与终点的位置关系,设定寻路方向的优先级顺序,减少不合理方向上的冗余规划计算量;其次,修改到达终点的判断条件,可在轨迹规划时实现路径的跳跃;再次,针对 A* 算法轨迹规划效率低的问题,提出自适应步长策略;最后,针对内存占用大,以及面对大地图时可能出现的内存溢出问题,提出了八方向搜索法.实验结果表明,相较于原始的 A* 算法,改进的 A* 算法在轨迹规划效率上获得了极大的提升,同时内存占用大的问题也得到了很好的解决.

关键词

路径规划;A* 算法;自适应步长

中图分类号 TP242

文献标志码 A

收稿日期 2023-02-18

资助项目 江苏省第五期"333 工程"科研项目(BRA2020067)

作者简介

付雄,男,硕士生,研究方向为自动驾驶.949082297@qq.com

李涛(通信作者),男,博士,教授,研究方向为抗干扰控制.litaojia@163.com

1 南京信息工程大学 自动化学院,南京,210044

2 南京信息工程大学 大气环境与装备技术协同创新中心,南京,210044

0 引言

路径规划不仅在机器人运动方面[1-2]有重要的作用,在无人机自主避障飞行[3]、游戏中自动引路系统[4]等方面也有广泛应用.路径规划的目的是在充满复杂障碍物的环境中找到一条从起点到终点的最佳路径[5].关于机器人路径优化的问题,国内外许多学者进行了大量的研究.本文主要针对机器人路径规划效率展开研究.

现有的路径规划算法有很多种,包括 A* 算法[6]、蚁群算法[7]、D* 算法[8]、RRT 算法[9]、人工势场法[10]、模拟退火法[11]等.蚁群算法是一种仿生算法,它模拟大自然蚂蚁寻找食物的过程,具有较强的鲁棒性特征,但存在计算量大、求解时间长等问题.单纯的 D* 算法并不考虑目标方向问题,通过暴力搜索的方式寻得最优解,但在环境地图增大时,其计算量也会急剧增加.A* 算法是一种基于启发函数的路径规划算法,也是目前常用的机器人路径规划算法.A* 算法在求解过程中,会不断将当前节点及周围 8 个邻近点添加到封闭列表和开放列表中进行代价评估,因此会导致大量非必要的节点加入到计算中,存在计算效率低下等问题.Goldberg 等[12]研究了评估函数的启发强度,提升了算法的计算速度,但算法严重占用内存;Harabor 等[13]提出一种跳点搜索法,引入强迫邻居的概念,大幅度提高了运算速度;辛煜等[14]提出了无限邻域的思想,使得搜索路径不再局限于 8 个方向,该方法虽然缩短了路径的长度,但增加了计算量;Chaari 等[15]提出一种松弛 A* 算法,与原始 A* 算法相比,计算时间得到有效减少;张庆等[16]提出一种融合 JPS(跳点搜索)策略的改进算法.

根据上述算法的改进思想及 A* 算法寻路的底层逻辑,本文提出一种改进的 A* 算法.在路径搜索方向上根据当前点与终点的夹角,设定寻路方向的优先级关系;接着修改到达终点的判断条件,并根据当前点、障碍物及终点的位置关系给出自适应步长的求解方法;当自适应步长的长度大于 1 时,其搜索方向超过 8 个方向,为了减少冗余评估节点的介入,提出固定八方向搜索规则.最后通过仿真及实验验证了改进算法的有效性.

1 问题描述

1.1 环境表示

栅格法是路径规划中最常用的建模方法,由 Elfes 等[17]提出.它

以九宫格的方式将环境地图等分,如图 1 所示,白色代表可通行的地方,记录为“0”,黑色代表不可通行的地方,记录为“1”,这样规定之后,地图存储的信息量少,方便创建和维护.栅格法将环境地图用二维矩阵表示后,路径规划问题就转变为处理二维矩阵问题.

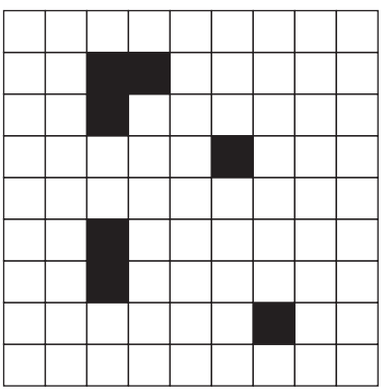


图 1　栅格地图

Fig. 1　Grid map

1.2　原始 A* 算法

A* 算法是启发式路径规划算法,适用于已知全局障碍物信息的路径轨迹求解,其路径搜索方向有 8 个方向,如图 2 所示.

A* 算法通过一个代价评估函数 $f(N)$ 来求解,其中,N 为当前节点,坐标为 (x_N,y_N),E 为终点,坐标为 (x_E,y_E).从起点出发,朝着终点拓展,再利用 $f(N)$ 计算出每个节点的代价值:

$$f(N)=g(N)+h(N). \tag{1}$$

式中:$g(N)$ 为代价函数,描述的是起点到当前点的实际距离;$h(N)$ 为启发函数,描述的是当前点到终点的估计距离.根据不同情况,$h(N)$ 可使用曼哈顿距离、切比雪夫距离或欧几里得距离,本文采用曼哈顿距离,其公式为

$$h(N)=|x_N-x_E|+|y_N-y_E|. \tag{2}$$

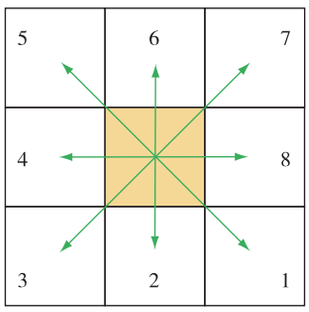


图 2　路径搜索方向

Fig. 2　Diagram of search directions for pathfinding

A* 算法在进行路径求解时会预先定义两个集合,一个为封闭列表,另一个为开放列表,分别用来存放已经遍历过和将要遍历的节点.首先把起点加入到封闭列表中,然后将开放列表中 $f(N)$ 最小的点,加入到封闭列表中,把 N 的周边节点加入到开放列表中,重复以上步骤直到终点也在开放列表中,算法结束.

2　算法改进

2.1　设定寻路方向的优先级关系

改进的 A* 算法根据终点与当前点的夹角 θ,设定算法寻路的搜索方向.如图 3 所示,设定地图的左上角为原点 O,向右为 x 轴,向下为 y 轴,当前点坐标为 (x_N,y_N),终点坐标为 (x_E,y_E),则夹角为

$$\theta=\arctan\left(\frac{y_E-y_N}{x_E-x_N}\right),\theta\in\left[-\frac{\pi}{2},\frac{\pi}{2}\right]. \tag{3}$$

根据 θ 的值,对照图 2 方向给出方向表 1.

例如图 3 左图,终点坐标为(7,7),当前点坐标为(4,4),夹角 $\theta=\arctan\left(\frac{7-4}{7-4}\right)=0.785\ 398$,$\theta\in$

表 1　θ 与算法寻路方向关系

Table 1　Relationship between θ and algorithm's pathfinding directions

θ 取值	方向	θ 取值	方向
$\left(\frac{\pi}{8},\frac{3\pi}{8}\right),\sin\theta>0$	“1”	$\left(\frac{\pi}{8},\frac{3\pi}{8}\right),\sin\theta<0$	“5”
$\left(\frac{3\pi}{8},\frac{\pi}{2}\right]\cup\left[-\frac{\pi}{2},-\frac{3\pi}{8}\right),\sin\theta>0$	“2”	$\left(\frac{3\pi}{8},\frac{\pi}{2}\right]\cup\left[-\frac{\pi}{2},-\frac{3\pi}{8}\right),\sin\theta<0$	“6”
$\left[-\frac{3\pi}{8},-\frac{\pi}{8}\right),\sin\theta>0$	“3”	$\left[-\frac{3\pi}{8},-\frac{\pi}{8}\right),\sin\theta<0$	“7”
$\left[-\frac{\pi}{8},\frac{\pi}{8}\right],\cos\theta<0$	“4”	$\left[-\frac{\pi}{8},\frac{\pi}{8}\right],\cos\theta>0$	“8”

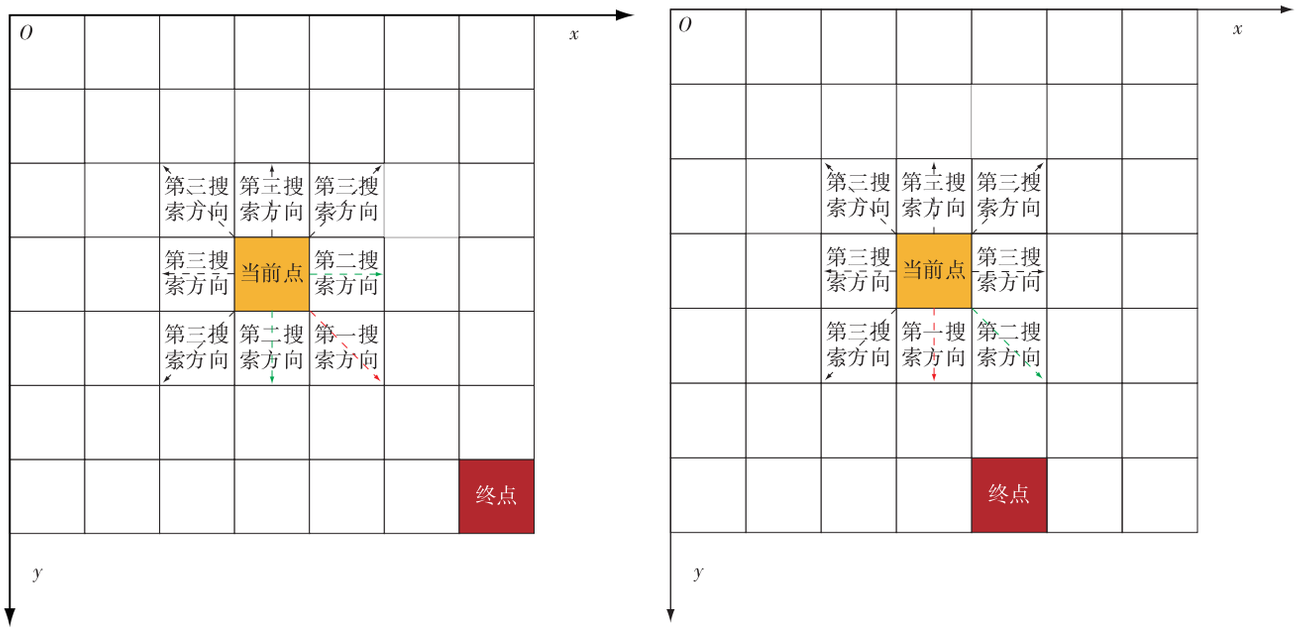


图 3　搜索方向示意图

Fig. 3　Schematic of search directions

$\left(\frac{\pi}{8},\frac{3\pi}{8}\right]$,且 $\sin\theta>0$,参照表 1 得出第一搜索方向为“1” 方向.例如图 3 右图,终点向左平移两格后夹角 $\theta=\arctan\left(\frac{7-4}{5-4}\right)=1.249\ 05$,其中 $\theta\in\left(\frac{3\pi}{8},\frac{5\pi}{8}\right]$,且 $\sin\theta>0$,参照表 1 得出第一搜索方向为“2” 方向.

注 1　设定寻路方向的优先级关系的目的是使得算法在寻路运算中,有针对性地沿着最佳路径搜索,减少冗余计算.

2.2　修改到达终点的判断条件

原始 A* 算法到达终点的条件为终点在最后一个节点的开放列表中.本文提出新的结束条件,当终点在当前节点的第一搜索方向上且无障碍物,即可结束路径搜索工作,如图 4 所示.

注 2　对比原始算法到达终点的规则,本文规则可以实现最后一步的跳跃,减少算法运算量.若终点在起点的第一搜索方向上且无障碍物,则可直接得出路径轨迹,无需进入算法迭代.

2.3　加入自适应步长策略

针对 A* 算法在路径轨迹计算方面耗时长的问题,加入自适应步长策略,如图 5 所示,绿色栅格为起点,红色栅格为终点,灰色栅格为参与评估节点.

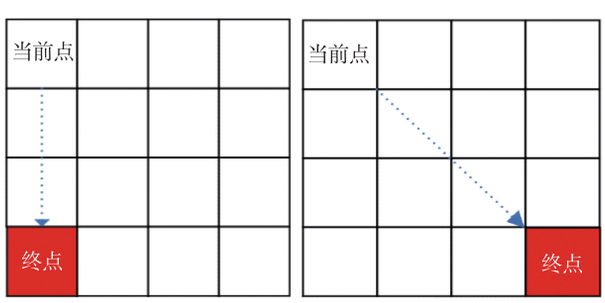


图 4　迭代结束示意图

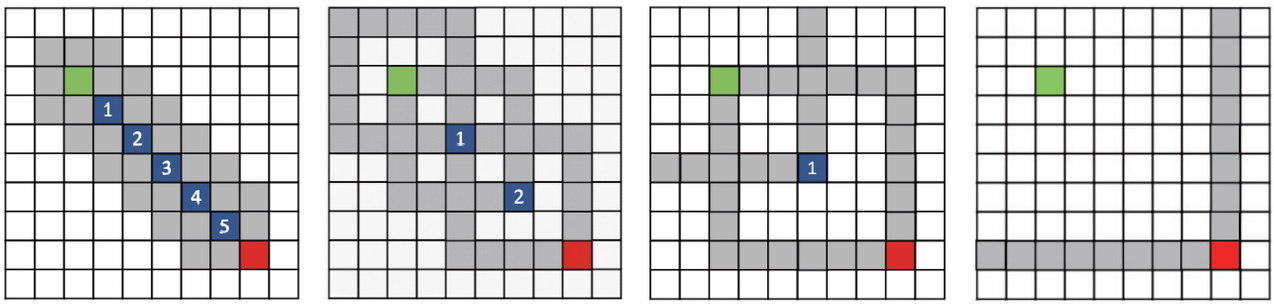
Fig. 4　Schematic of the end of iteration

当步长为 1 格时,如图 5a 所示需要 6 步可以到达终点,此时算法迭代次数为 6 次;当步长为 2 格时,如图 5b 所示,此时算法迭代次数为 3 次;当步长为 3 格时,如图 5c 所示,此时算法迭代次数为 2 次;当步长为 6 格时,如图 5d 所示,此时算法迭代次数为 1 次.当步长为其他数值时会陷入震荡,导致收敛速度慢,或者无法到达终点.例如,当步长为 4 格时,会陷入震荡无法到达终点.

本文提出的自适应步长策略,其步长的长度是根据当前节点、障碍物及终点的关系决定的,分以下两种情况讨论.

情况 1:节点第一搜索方向上不存在障碍物

如图 6a 所示,当前点 N 与终点 E 的夹角 $\theta=\arctan\left(\frac{5}{3}\right)=1.030\ 38$,$\theta\in\left(\frac{\pi}{8},\frac{3\pi}{8}\right]$,且 $\sin\theta>0$,



a. 步长为1　b. 步长为2　c. 步长为3　d. 步长为6

图 5　不同步长示意图

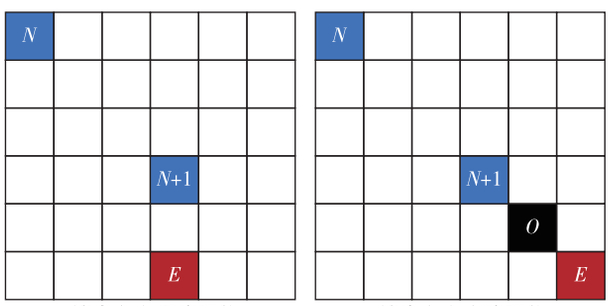
Fig. 5　Schematic of varying step sizes

选择“1”方向为第一搜索方向.当沿第一搜索方向拓展到 $N+1$ 节点时,由于其横坐标与终点横坐标相同(即“1”方向已不再是第一搜索方向),故停止拓展.N 的自适应步长 $\lambda=\min\{|x_E-x_N|,|y_E-y_N|\}$,其为当前点与终点横坐标差值的绝对值和纵坐标差值的绝对值中较小的那个.

情况 2:节点搜索方向上存在障碍物

如图 6b 所示,当前点 N 与终点 E 的夹角 $\theta=\arctan\left(\frac{5}{5}\right)=0.785\ 398,\theta\in\left(\frac{\pi}{8},\frac{3\pi}{8}\right]$,且 $\sin\theta>0$,选择“1”方向为第一搜索方向.从当前点 N 拓展至终点 E 时,记第一搜索方向上的第一个障碍物为 O,坐标为 (x_O,y_O),则当前点 N 的自适应步长 $\lambda=\max\{|x_S-x_O|,|y_S-y_O|\}$,其描述的是当前点与障碍物横坐标差值的绝对值和纵坐标差值的绝对值中较大的那个.将节点 $N+1$ 更新为当前点,坐标为 $(x_N+\lambda,y_N+\lambda)$,由于该节点的第一搜索方向为不可通行状态,故在此处按照原始 A* 算法寻路.

注 3　合适的步长会减少迭代次数,极大地缩减算法的运算耗时,特别是在大地图环境中或障碍物稀疏的地图中尤为明显.当环境地图足够大时,原始算法会出现内存溢出问题,导致机器人宕机.



a. 搜索方向无障碍物　b. 搜索方向有障碍物

图 6　自适应步长选取示意图

Fig. 6　Schematic of adaptive step size selection

2.4　八方向搜索法

步长只影响迭代次数,不会大幅度减少开放列表中的评估节点个数,所以占用内存大的问题依然存在.本文提出固定八方向搜索法,即步长增加时搜索方向数保持不变,这与文献[14]的无限邻域思想是相反的.固定八方向原理如图 7 所示.

对比图 5 与图 7 中评估节点数可以得出表 2 的结果.

注 4　评估节点的数量不仅会占用大量内存,同

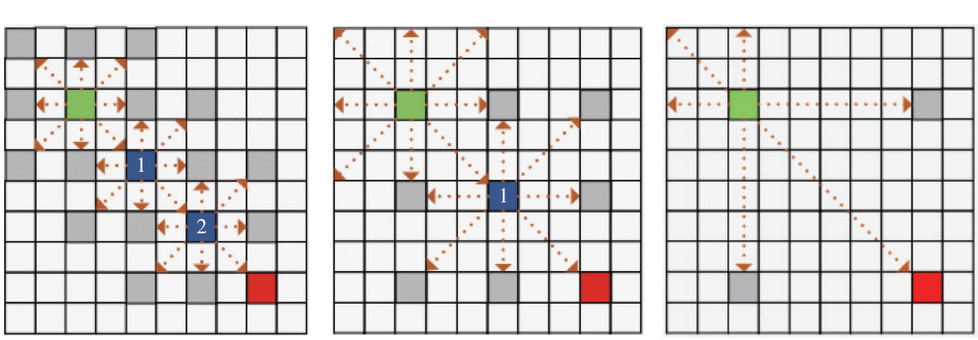


图 7　固定八方向搜索示意图

Fig. 7　Schematic of fixed eight-directional search

时也会影响算法寻路耗时,评估节点数的减少会进一步提升算法的轨迹规划效率.

表 2　原始算法评估节点数与改进的评估节点数

Table 2　Number of evaluation nodes for original A* algorithm and the improved A* algorithm

步长/格	原始节点数	改进后节点数	减少比例/%
2	42	18	57.14
3	33	8	75.76
6	17	3	82.35

2.5　算法流程

算法流程(图 8)如下:

Step 1:根据当前点和终点的位置关系,计算第一搜索方向.

Step 2:根据自适应步长策略,选取步长数.

Step 3:根据到达终点条件判断是否到达终点,如果是,进入 Step 5,如果不是,则进入 Step 4.

Step 4:按照 A* 算法规则进行迭代,并返回 Step 1.

Step 5:输出路径轨迹.

Step 6:算法结束.

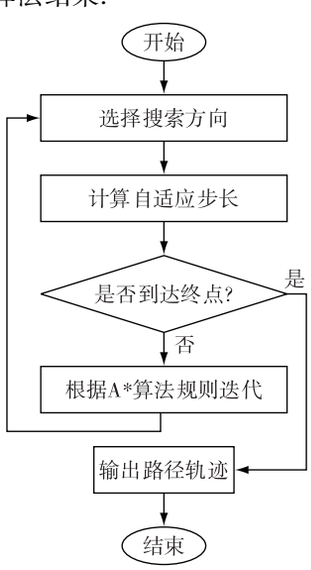


图 8　算法流程

Fig. 8　Flow of the improved A* algorithm

以下给出算法的简易例程,如图 9 所示.开始时 $N=0$,起点为当前点,并将起点放入封闭列表中,终点与当前点的夹角 $\theta=\arctan\left(\frac{9}{9}\right)=0.785\ 398,\theta\in\left(\frac{\pi}{8},\frac{3\pi}{8}\right]$,且 $\sin\theta>0$.根据表 1,得出“1”方向为第一搜索方向,代价评估函数 $f(0)=18$.因该方向上第 3 格为障碍物,根据 2.3 节中的情况 2,计算自适应步长 $\lambda=2$,更新 $N=1$ 为当前节点,并将其放入封闭列表中,代价评估函数 $f(1)=2\sqrt{2}+12$.在节点 $N=1$ 处,由于其第一搜索方向不可通行,按照原始 A* 算法寻路,将邻域节点放入开放列表中,根据式(1)可得开放列表集合为:$\{f(2)=2\sqrt{2}+12,f(4)=3\sqrt{2}+12,f(5)=2\sqrt{2}+14,f(6)=3\sqrt{2}+14,f(7)=2\sqrt{2}+14,f(8)=3\sqrt{2}+12,f(9)=2\sqrt{2}+12\}$,选取其中最小的作为下一评估节点,即 $f(2)$ 与 $f(9)$.以节点“9”作为当前点拓展时,根据 A* 算法不将已经评估过的节点重新纳入评估系统的规则,即开放列表中只有 $f(10)=3\sqrt{2}+12$.以节点“2”作为当前点拓展时,夹角 $\theta=\arctan\left(\frac{6}{5}\right)=0.876\ 058,\theta\in\left(\frac{\pi}{8},\frac{3\pi}{8}\right]$,且 $\sin\theta>0$,则“1”方向作为第一搜索方向.根据 2.3 节中情况 1,计算出自适应步长 $\lambda=5$,此时 $f(3)=7\sqrt{2}+2$,由于 $f(3)<f(10)$,故对节点“9”进行剪枝处理,舍去该节点及其分支,并将节点“2”放入封闭列表中.更新节点“3”为当前点,由于满足到达终点的判断条件,故将节点“3”与终点放入封闭列表中,此时封闭列表中有:{起点,$N=1$,$N=2$,$N=3$,终点},其连线为路径轨迹,算法迭代结束.

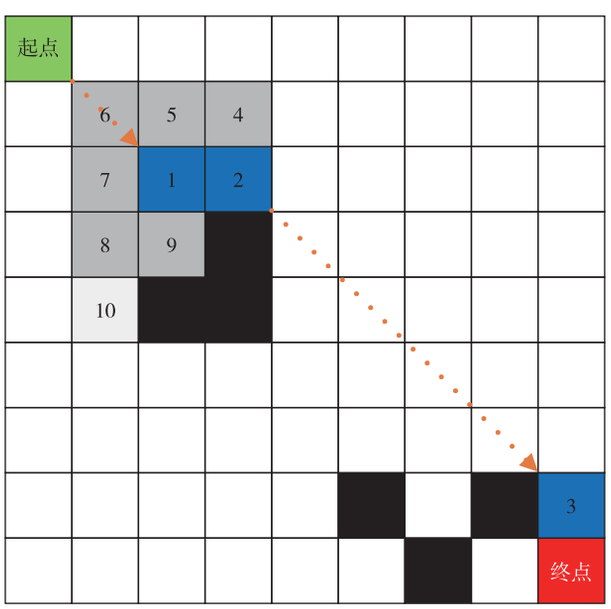


图 9　算法简易例程

Fig. 9　An example of the improved A* algorithm for pathfinding

3 仿真及实验验证

3.1 仿真分析

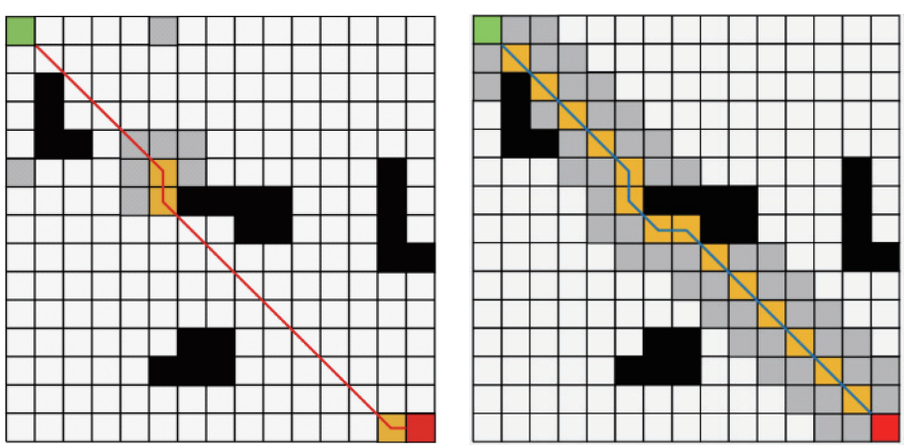
为了验证算法的效果,本文针对原始 A* 算法和改进后的算法进行了测试实验,可验证本文算法的正确性.系统 CPU 为 i5-8257U,操作系统为 Windows 10,主频为 1.4 GHz,运行内存为 8 GB,在主机上运行 MATLAB 2021a 进行仿真.

如图 10 所示,在 15×15 的栅格地图上进行仿真实验,绿色为起点,红色为终点,黑色为障碍物,黄色为路径节点,灰色为路径计算时参与的评估节点,可以看出相同环境下两种算法生成的路径轨迹区别不大,但改进的 A* 算法在规划路径轨迹时参与的路径节点和评估节点数要远少于原始的 A* 算法.

由于机器人的编程语言多为 C++语言,为进一步验证算法的效果,本文在 C++语言环境下做了大地图的仿真测试,实验环境均为障碍物占比率为 1% 的地图.在 250×200 的地图中,其测试结果如图 11a 所示;在 400×250 的地图中,其测试结果如图 11b 所示.其中,黑色为障碍物,空白区域为可通行空间,红色为改进算法的路径轨迹,蓝色为原始 A* 算法的路径轨迹.

表 3 给出了原始算法与改进算法在不同尺寸的栅格地图中路径规划用时及节点个数的数据对比.路径规划算法的评价指标有时间和路径长度两种,本文主要以时间作为评价指标.

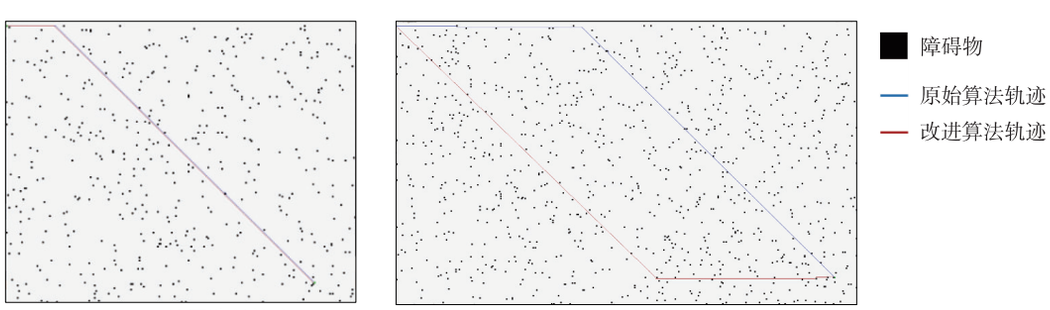
同时本文以障碍物占比率为变量进行多次仿真实验,结果如图 12 所示.可以看出障碍物的数量变化对原始 A* 算法的影响很小,对改进的算法影响极大.由于改进算法的自适应步长选择是根据当前节点与障碍物的位置关系决定的,所以障碍物占比率低的地图,路径规划效率提升明显.



a. 改进A*算法　　b. 原始A*算法

图 10　路径轨迹对比

Fig. 10　Comparison of simulated path trajectories



a. 250×200地图路径轨迹　　b. 400×250地图路径轨迹

图 11　路径轨迹对比

Fig. 11　Comparison of path trajectories

表 3　原始 A* 算法与改进 A* 算法结果数据对比

Table 3　Comparison of results between original A* algorithm and the improved A* algorithm

地图大小	平均路径节点			平均用时		
	改进 A* 算法/个	原始 A* 算法/个	性能提升/%	改进 A* 算法/ms	原始 A* 算法/ms	性能提升/%
15×15	3	14	78.57	0.418	1.795	76.71
250×200	12	242	95.41	1.594	203.545	99.22
400×250	15	383	96.08	1.977	441.043	99.55

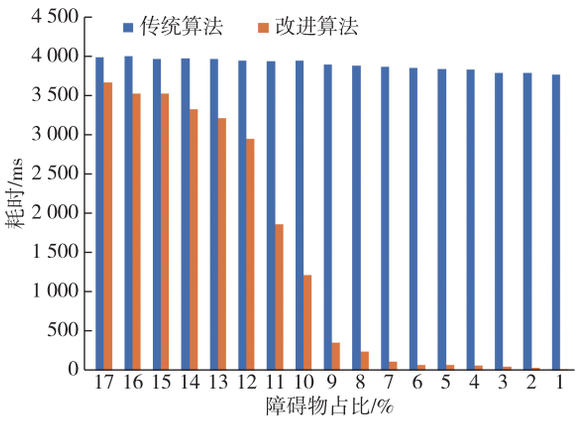


图 12　障碍物占比与耗时关系

Fig. 12　Diagram of obstacle ratio and time consumption

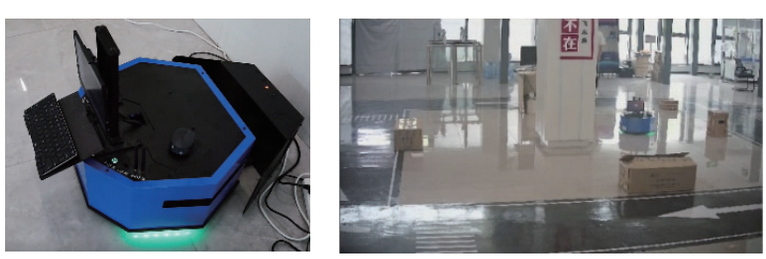
3.2　实验分析

为了验证算法的实际效果,本文将改进的算法作为路径规划器插件应用在实物机器人上,实物机器人采用的启智底盘,如图 13a 所示.实验场地为 18 m×12 m 的密闭空间,室内放置了桌子、纸箱等障碍物,如图 13b 所示.场地中以同一起点和终点做了两次路径规划任务,其路径轨迹结果在 Rviz 显示如图 14 所示,可以看出两种算法规划的路径轨迹近似,但本文改进的算法在用时上远少于原始算法.同时表 4 给出 4 种算法在实验环境下的路径规划耗时对比,可以看出文献[5]算法的路径虽然更优,但耗时较长,在大地图时并不适用,文献[16]与本文算法路径长度近似,但本文算法规划用时更短.

表 4　不同算法寻路数据对比

Table 4　Comparison of pathfinding data between different algorithms

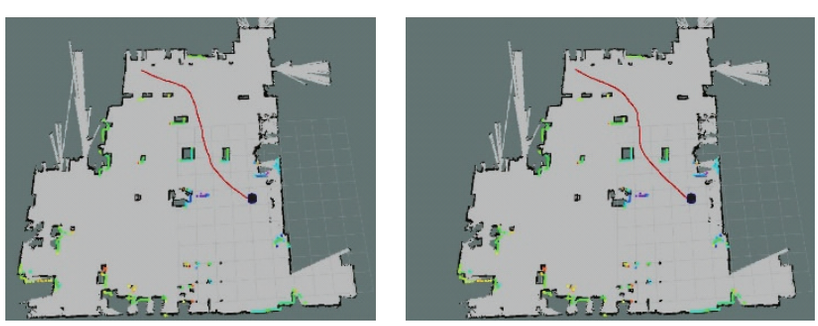
算法	规划用时/ms	路径长度
A*	152.25	11.32
文献[5]	95.92	10.75
文献[16]	12.18	11.38
本文	4.94	11.29



a. 机器人　　b. 试验场地

图 13　机器人及场地

Fig. 13　Pictures of robot and test field



a. 改进A*算法　　b. 原始A*算法

图 14　路径规划结果

Fig. 14　Path planning results

4 结束语

本文针对 A* 算法路径规划耗时长的问题,加入了自适应步长策略;针对内存溢出问题,提出八方向搜索法,并通过设定搜索方向的优先级顺序,进一步减少评估节点,有效地降低了内存使用率.改进后的算法能更好地满足机器人路径规划任务,并通过仿真和实际实验验证了该方法的有效性.

参考文献

References

[1] 李晓旭,马兴录,王先鹏.移动机器人路径规划算法综述[J].计算机测量与控制,2022,30(7):9-19
LI Xiaoxu, MA Xinglu, WANG Xianpeng. A survey of path planning algorithms for mobile robots[J]. Computer Measurement & Control,2022,30(7):9-19

[2] 李佳伟,林娜,池荣虎.基于数据驱动高阶学习律的轮式移动机器人轨迹跟踪控制[J].南京信息工程大学学报(自然科学版),2021,13(1):66-72
LI Jiawei, LIN Na, CHI Ronghu. Data driven high order learning control for path tracking of wheeled mobile robots[J]. Journal of Nanjing University of Information Science & Technology (Natural Science Edition),2021,13(1):66-72

[3] 陈亚青,张智豪,李哲.无人机避障方法研究进展[J].自动化技术与应用,2020,39(12):1-6
CHEN Yaqing, ZHANG Zhihao, LI Zhe. Research progress of obstacle avoidance methods for unmanned aerial vehicles[J]. Techniques of Automation and Applications,2020,39(12):1-6

[4] 刘梓良.面向游戏地图的寻径算法的研究与实现[D].南京:东南大学,2016
LIU Ziliang. Research and implementation of path finding algorithm for game map [D]. Nanjing: Southeast University,2016

[5] 岳高峰,张萌,沈超,等.移动机器人导航规划的双向平滑 A-star 法[J].中国科学(技术科学),2021,51(4):459-468
YUE Gaofeng, ZHANG Meng, SHEN Chao, et al. Bidirectional smoothing A-star method for navigation planning of mobile robots[J]. Scientia Sinica (Technologica),2021,51(4):459-468

[6] 支琛博,张爱军,杜新阳,等.改进 A* 算法的移动机器人全局路径规划研究[J].计算机仿真,2023,40(2):486-491
ZHI Chenbo, ZHANG Aijun, DU Xinyang, et al. Research on global path planning of mobile robot based on improved A* [J]. Computer Simulation, 2023, 40(2): 486-491

[7] 李燕,季建楠,沈葭栎,等.基于改进蚁群算法的移动机器人路径规划方法[J].南京信息工程大学学报(自然科学版),2021,13(3):298-303
LI Yan, JI Jiannan, SHEN Jiali, et al. Mobile robot path planning based on improved ant colony algorithm[J]. Journal of Nanjing University of Information Science & Technology (Natural Science Edition), 2021, 13(3): 298-303

[8] Ciesielski K C, Falcão A X, Miranda P A V. Path-value functions for which Dijkstra's algorithm returns optimal mapping [J]. Journal of Mathematical Imaging and Vision, 2018, 60(7): 1025-1036

[9] Xie B Y, Zhao J, Liu Y. Fault tolerant motion planning of robotic manipulators based on a nested RRT algorithm [J]. The Industrial Robot, 2012, 39(1): 40-46

[10] 徐小强,王明勇,冒燕.基于改进人工势场法的移动机器人路径规划[J].计算机应用,2020,40(12):3508-3512
XU Xiaoqiang, WANG Mingyong, MAO Yan. Path planning of mobile robot based on improved artificial potential field method [J]. Journal of Computer Applications, 2020, 40(12): 3508-3512

[11] 王星童,吴林鸿,赵启宇,等.粒子群-快速模拟退火算法在路径规划中的应用[J].信息技术与信息化,2021(6):13-16
WANG Xingtong, WU Linhong, ZHAO Qiyu, et al. Application of particle swarm optimization-fast simulated annealing algorithm in path planning[J]. Information Technology & Informatization, 2021(6): 13-16

[12] Goldberg A V, Harrelson C. Computing the shortest path: A* search meets graph theory[C]//Annual ACM-SIAM Symposium on Discrete Algorithms. New York, USA: ACM, 2005: 156-165

[13] Harabor D, Grastien A. Online graph pruning for pathfinding on grid maps[C]//Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. August 7-11, 2011, San Francisco, California. New York: ACM, 2011: 1114-1119

[14] 辛煜,梁华为,杜明博,等.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 Yu, LIANG Huawei, DU Mingbo, et al. An improved A* algorithm for searching infinite neighbourhoods[J]. Robot, 2014, 36(5): 627-633

[15] Chaari I, Koubaa A, Bennaceur H, et al. Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments[J]. International Journal of Advanced Robotic Systems, 2017, 14(2): 1-27

[16] 张庆,刘旭,彭力,等.融合 JPS 和改进 A* 算法的移动机器人路径规划[J].计算机科学与探索,2021,15(11):2233-2240
ZHANG Qing, LIU Xu, PENG Li, et al. Path planning for mobile robots based on JPS and improved A* algorithm [J]. Journal of Frontiers of Computer Science and Technology, 2021, 15(11): 2233-2240

[17] Elfes A, Moravec H. Automatic occupancy grid generation from multiple sensory measurements[C]//Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems, 1985: 576-584

Path optimization of A* algorithm based on adaptive step size strategy

FU Xiong[1] LI Tao[1,2]

1 School of Automation,Nanjing University of Information Science & Technology,Nanjing 210044,China

2 Collaborative Innovation Center of Atmospheric Environment and Equipment Technology,
Nanjing University of Information Science & Technology,Nanjing 210044,China

Abstract To address the problem of long time-consuming and large memory consumption of A* algorithm in solving path trajectory,this paper proposes an improved A* algorithm based on adaptive step size.First,the priority order of the search direction was set according to the position relationship between the current point and the end point,with the purpose to reduce the redundant planning calculation on unreasonable directions. Second, the judgment condition for reaching the end point was modified to achieve path jumping during path planning.Then an adaptive step size strategy was proposed to improve the efficiency of A* algorithm in path planning. Finally, an eight-directional search approach was proposed to address the issues of large memory usage and possible memory overflow when facing large maps.Experimental results show that compared with original A* algorithm,the improved A* algorithm greatly improves the efficiency of path planning,and solves the problem of large memory usage.

Key words path planning;A* algorithm;adaptive step size