文章编号：2095-2163(2021)11-0014-07 中图分类号：TP242.6 文献标志码：A

融合改进 A* 与 DWA 的轮式机器人路径规划

游向荣，方海龙，唐瑞东
（武汉科技大学 汽车与交通工程学院，武汉 430065）

摘 要：针对轮式机器人在路径规划中存在的问题，提出融合改进 A* 和 DWA 的轮式机器人路径规划方法。首先针对 A* 算法冗余点多、耗时长、拐点多等问题，提出双向 A* 和跳点算法相结合来提升全局路径规划效率；其次针对 DWA 寻找的路径非最优以及无法应变环境的问题，选取全局最优路径上优化后的跳点作为 DWA 的关键点，引入关键点评价子函数和自适应速度权值。融合改进的 A* 和 DWA 算法，既保证了全局路径最优，还可根据障碍物数量和分布，调整速度权值，兼顾速度和安全性。经在 MATLAB 仿真环境中验证：融合改进 A* 和 DWA 的轮式机器人路径规划方法与传统 A* 算法相比，大幅减少了规划时长和拓展节点数量，效率明显提升；与 DWA 算法相比，加入全局路径关键点和自适应速度权值，能遵循全局最优路径以及降低了运行时间，提升效率。

关键词：双向 A*；JPS；DWA；自适应速度权值；路径规划

Path planning of wheeled robot based on improved A* and DWA

YOU Xiangrong, FANG Hailong, TANG Ruidong
(College of Automobile and Traffic Engineering, Wuhan University of Science and Technology, Wuhan 430065, China)

【Abstract】 Aiming at the problems of path planning for wheeled robots, a path planning method for wheeled robots based on improved A* and DWA is proposed. Firstly, aiming at the problems of redundant points, time-consuming and inflection points of A* algorithm, a combination of bidirectional A* and hop point algorithm is proposed to improve the efficiency of global path planning; Secondly, aiming at the problem that the path searched by DWA is not optimal and cannot adapt to the environment, the optimized jump point on the global optimal path is selected as the key point of DWA, and the key point evaluation sub function and adaptive speed weight are introduced. The fusion of improved A* and DWA algorithm not only ensures the global optimal path, but also adjusts the speed weight according to the number and distribution of obstacles, taking into account the speed and safety. In the MATLAB simulation environment, it shows that: compared with the traditional A* algorithm, the planning time and the number of expansion nodes are greatly reduced, and the efficiency is significantly improved; Compared with DWA algorithm, adding global path key points and adaptive speed weights can follow the global optimal path, reduce the running time and improve the efficiency.
【Key words】 Bidirectional A*; JPS; DWA; Adaptive velocity weights; Path planning

0 引 言

路径规划问题最早起源于机器人从起始点到终点寻找出一条最优的无碰撞路径。机器人路径规划环境分为静态环境和动态环境。静态环境是在已知先验地图模型下，在静态障碍物中规划一条全局路径；动态环境是指通过机器人搭载的传感器感知周围环境中的动态障碍物，从而进行安全避障[1]。

全局路径规划算法有：Dijkstra、A*、RRT、蚁群算法等[2-4]；局部路径规划的算法有：DWA、人工势场、D*、D* lite 等[5-9]。其中，A* 算法存在着冗余点多、耗时长、拐点多等问题。文献[10]中提出，通过改进算法的启发函数 $h(n)$，并加入车身轮廓代价和障碍物代价，最后使用贝塞尔曲线拟合转折点，使规划的路径更加平滑、合理，但是依然没有提升遍历效率。文献[11-12]中引入双向 A* 搜索机制，缩短寻路时间，但却增加了对周边节点的遍历量。由于 DWA 算法容易陷入局部最优，卞永明[13]等人取出 A* 全局路径中的关键航迹点，以关键航迹点到待评价轨迹的距离作为依据，定义新的评价子函数，获得新型 DWA 评价函数。因此可以有效避开“C”形障碍物，跳出局部最优。

综上，针对 A* 和 DWA 算法存在的缺点，提出双向 A* 和 JPS 算法融合。通过 JPS 算法筛选跳点的方法，选出关键节点。其目的，一是实现全局路径规划中远距离的跳跃，二是作为 DWA 算法的局部

作者简介：游向荣(1973-)，男，硕士，高级工程师，主要研究方向：汽车电子、计算机控制、工程机械智能化；方海龙(1996-)，男，硕士研究生，主要研究方向：计算机电子控制；唐瑞东(1997-)，男，硕士研究生，主要研究方向：嵌入式控制。
通讯作者：游向荣 Email：1044696483@qq.com
收稿日期：2021-08-17

关键点，作为关键点评价子函数的依据。此外，根据周边障碍物数量和分布，自动调节速度权值，在保证安全的基础上提升效率。

1　改进 A* 算法

1.1　传统 A* 算法

A* 寻路算法，是将周边环境处理为栅格地图（如图 1 所示）。白色格子表示可行区域，黑色表示不可达到区域，即障碍物或边界。

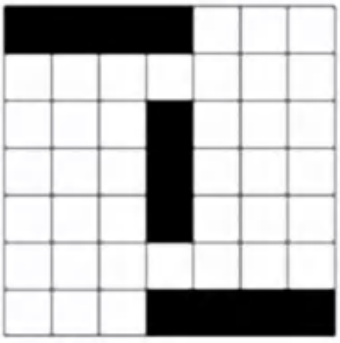


图 1　栅格地图

Fig. 1　Grid map

A* 算法的核心是对周边 8 个方向的节点进行处理，选出最优节点后反复迭代，直到找到终点。其判断依据是根据代价函数 $F(n)$，表示为式（1）：

$$F(n) = G(n) + H(n) \tag{1}$$

式中：$F(n)$ 表示起始点经当前节点 n 到终点的估计代价函数；$G(n)$ 表示起始点经到当前节点 n 的实际距离代价；$H(n)$ 表示当前节点 n 到终点的估计距离代价。一般采用曼哈顿或欧式距离。

1.2　JPS 算法

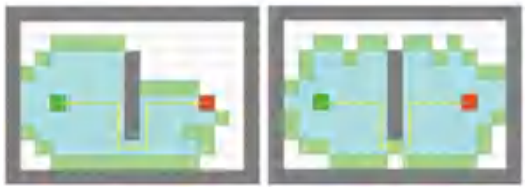
跳点搜索（JPS）是对 A* 搜索算法的优化，通过图裁剪来减少搜索过程的对称性，能让搜索在网格上直线长“跳”，而不是普通 A* 的小步移动[14]，相当于对栅格地图预处理，删除无用节点，保留关键节点作为跳点，再使用 A* 算法。在这个过程中，减少加入 openlist 的节点，降低对 openlist 的处理时间。

JPS 算法的关键在于筛选出跳点，为考虑轮式机器人自身尺寸，不允许经过对角线到达强迫邻居，需沿着父节点的方向再进一步，避免对角线穿越造成碰撞危险。

1.3　A* 算法优化

1.3.1　双向跳点 A*

传统 A* 算法为单向搜索，效率低，因此有学者提出双向 A* 算法。传统 A* 和双向 A* 算法寻路如图 2 所示。



（a）传统 A*　　（b）双向 A*

图 2　传统 A* 和双向 A*

Fig. 2　The algorithms of Traditional A* and bidirectional A*

由文献[15]可知，在该种情况下寻路时间减少，但是拓展节点数量反而增多。由于双向 A* 算法在起始点和终点之间存在障碍物时，易形成并行的搜索区间，导致了遍历节点增加。

JPS 算法可以大幅减少加入 openlist 的数量，因此将双向 A* 算法与 JPS 算法相结合。起始点到终点，终点到起始点，两个方向交替进行，即建立 openlist1 和 openlist2。当正向搜索到跳点时，加入 openlist1 并停止正向搜索，然后反向搜索到跳点时，同样加入 openlist2 中并停止反向搜索，再切换到正向搜索，直到正反向搜索将同一个跳点加入两个 openlist 中，表示路径被找到。

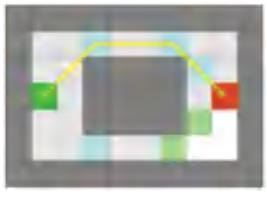


图 3　双向跳点 A* 算法

Fig. 3　Bidirectional jump point A* algorithm

由图 3 可知，双向跳点 A* 算法寻找的路径有两条，但由于路径长度相等，仅选择其中一条。这种情况不仅没有提高寻路效率，同样增加了遍历的节点数量。造成该现象的原因是：正反向搜索均是以起始点和终点作为路径搜索的目标点，在寻找路径的过程中出现了代价相同的节点，因此找到两条代价相同的路径。

解决上述问题的方案，是不再将起始点和终点作为反向搜索和正向搜索的目标点，而是将双向跳点寻路过程中产生跳点的前一个跳点作为搜索的目标点。一是为保证不再出现两条代价相同的路径；二是在寻路过程中对角线方向搜索跳点时，出现的跳点的数量可能大于 1。若以当前跳点作为搜索的目标点，则寻找的路径不一定最优，且增加处理时间；三是在寻路过程中，正反向搜索时，路径已经重合的情况下，跳点不一定重合。改进双向跳点 A* 寻路过程如图 4 所示。

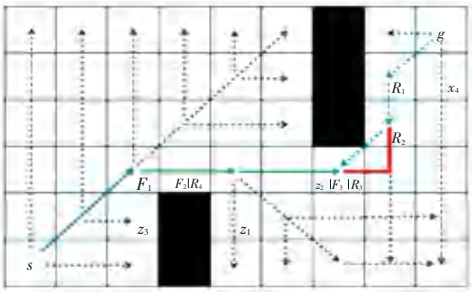


图 4 改进双向跳点 A* 算法

Fig. 4 Improved bidirectional jump point A* algorithm

图 4 中 s,g 分别为起始点和终点;$F_i(i = 1,2,\cdots)$ 和 $R_i(i = 1,2,\cdots)$ 分别为正反向搜索到的跳点;$z_i(i = 1,2,\cdots)$ 为强迫邻居;黑色虚线表示跳点寻路过程;绿色实线表示正向搜索路径;绿色虚线表示反向搜索路径;红色实线表示为了避免转角碰撞的微调路径。改进的双向跳点 A* 算法流程如图 5 所示。

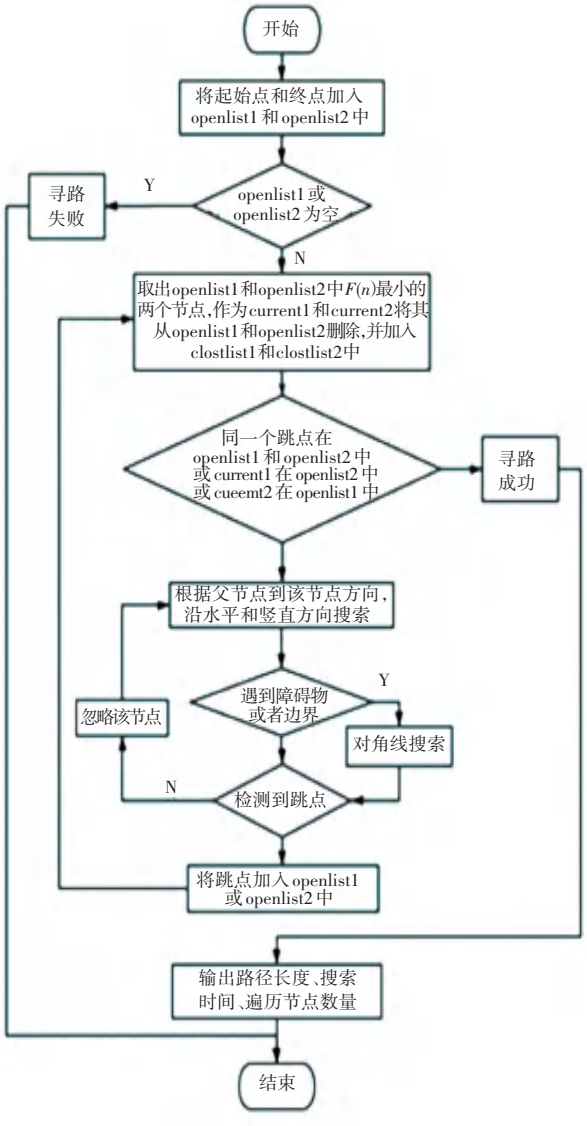


图 5 改进算法流程图

Fig. 5 Flow chart ofimproved algorithm

1.3.2 代价函数的改进

JPS 算法在筛选出跳点后,需采用公式(1)来评价出最优跳点。分析公式(1)可知,若当前节点一旦确定,影响 $F(n)$ 的只有 $H(n)$。当 $H(n) = 0$ 时,A* 算法蜕变成 Dijkstra 算法,加入 openlist 中的节点增多,效率降低;当 $H(n)$ 不为 0 但小于实际距离时,处理的节点同样较多;当 $H(n)$ 等于实际距离时为最优,但难以计算;当 $H(n)$ 大于实际距离时,处理的点数少、效率高,但不保证得到最优解,因此合适的代价函数至关重要。

针对该问题,不少学者提出相关方案。文献[10]中提出采用圆弧曲线对 $H(n)$ 进行估计,这在简单环境中是可行,因为简单环境中实际距离的取值在欧式距离和曼哈顿距离之间,但在复杂的环境中,存在着实际距离大于曼哈顿距离的情况。因此,本文结合双向跳点 A* 算法,提出动态 $H(n)$ 值的计算方法。该动态计算方法的思想是:双向跳点 A* 算法为双向同时搜索路径的算法,在寻路过程中正反向均会计算 $G(n)$ 和 $F(n)$ 值。传统的 $H(n)$ 是计算当前节点到终点的代价值,误差较大,如若将之改为当前节点到跳点的代价值和跳点到终点的代价值之和,则误差将会缩小,因此改进后的代价函数为:

$$\begin{aligned} F_1(n) &= G_1(n) + aH_1(n) + bG_2(n) \\ F_2(n) &= G_2(n) + aH_2(n) + bG_1(n) \end{aligned} \tag{2}$$

式中:F_1 表示正向搜索的总代价;F_2 表示反向搜索的总代价;G_1 表示起始点到正向搜索跳点的实际代价;G_2 表示终点到反向搜索跳点的实际代价;H_1 表示正向搜索的跳点到反向搜索跳点的估计代价;H_2 表示反向搜索跳点到正向搜索的跳点的估计代价;a 为跳点权重系数;b 为剩余代价权重系数。此外,$H(n)$ 的计算方法也需要调整,调整后的公式为:

$$H(n) = \frac{O(n) + M(n)}{2} \tag{3}$$

式中:$O(n)$ 表示欧式距离,$M(n)$ 表示曼哈顿距离。

1.4 路径平滑优化

双向跳点 A* 算法规划的路径依然存在着冗余点,路径转折过多,这是由 JPS 算法的特性决定的。针对 JPS 算法的缺陷,跳点优化步骤如下。

Step 1 获取双向跳点 A* 算法规划路径上的全部跳点,放入集合 $Q = \{q_i, 1 < i < n\}$;

Step 2 从 q_1 开始依次连接跳点,若中间存在障碍物,则该跳点保留;若中间不存在障碍物,则将此跳点删除,连接下个跳点,直到连至终点。删除后的跳点仅包含起始点、转折点和终点,将其加入集合 $U = \{u_j, 1 < j < m\}$。

Step 3　依次计算集合 U 中前后两点 u_j 和 u_{j+1} 形成的路径与最近障碍物的距离 d，并与安全距离 D 做判断。若 $d < D$，表明该条路径不安全，依次计算 u_j 和 u_{j+1} 之间跳点 q_i 三者之间的距离 L，在确保 u_j 与 q_i 以及 q_i 和 u_{j+1} 形成的两段路径与最近障碍物的距离 d 均大于 D 的前提下，取出 L 最小的跳点 qi，插入 u_j 和 u_{j+1} 之间，则新的集合为 $U = \{u_j, q_i, u_{j+1}, \cdots, u_{m+1}\}$。

Step 4　依次连接集合 U 中的节点，完成路径平滑优化。

2　改进 DWA 算法

2.1　传统 DWA 算法

DWA 算法是根据移动机器人自身机械特性，以及环境对速度的限制形成动态窗口进行速度采样，并模拟采样速度一定时间内 Δt 生成待评价轨迹。结合评价函数对待评价轨迹评分，再选择其中评分最高轨迹对应的采样速度作为最优速度[13]，当作移动机器人下一时刻速度。

2.1.1　轮式机器人运动模型

假设轮式机器人在 Δt 内做匀速直线运动，运动模型为：

$$\begin{bmatrix} x_{t+\Delta t} \\ y_{t+\Delta t} \\ \theta_{t+\Delta t} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & 0 \\ \sin\theta & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v\Delta t \\ v\Delta t \\ \omega\Delta t \end{bmatrix} \tag{4}$$

式中，$[x_{t+\Delta t}\ y_{t+\Delta t}\ \theta_{t+\Delta t}]^{\mathrm{T}}$ 是机器人在 $t + \Delta t$ 时刻的世界坐标系位姿；$[x_t\ y_t\ \theta_t]^{\mathrm{T}}$ 是机器人在 t 时刻的世界坐标系位姿；$[v\Delta t\ \ v\Delta t\ \ \omega\Delta t]^{\mathrm{T}}$ 是机器人在 Δt 时刻的位姿变化。

2.1.2　机器人速度约束

对机器人的速度约束需要考量 3 个方面的速度：一是机器人自身的最大、最小速度约束；二是机器人电机加减速度约束，即在时间间隔内，所能达到的最大最小速度；三是制动距离约束，即在撞到障碍物之前，机器人能以最大减速度将速度降至为 0。其约束公式如下：

$$V_m = \{(v,\omega) \mid v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\} \tag{5}$$

$$V_d = \begin{cases} (v,\omega) \mid v \in [v_c - v_b^* \Delta t, v_c + v_a^* \Delta t] \cap \\ \omega \in [\omega_c - \omega_b^* \Delta t, \omega_c + \omega_a^* \Delta t] \end{cases} \tag{6}$$

$$V_a = \begin{cases} (v,\omega) \mid v \leqslant \sqrt{2 \cdot dist(v,\omega) \cdot v_b^*}, \\ \omega \leqslant \sqrt{2 \cdot dist(v,\omega) \cdot \omega_b^*} \end{cases} \tag{7}$$

式中，v_c、ω_c 为当前速度；v_a、ω_a 为最大加速度；v_b、ω_b 为最大减速度；$dist(v, \omega)$ 表示 (v, ω) 对应轨迹距离障碍物的最近距离。

综上，动态窗口为上述 3 个约束的交集。

$$V_r = V_m \cap V_d \cap V_a \tag{8}$$

2.1.3　评价函数

对模拟轨迹的评价函数如公式(9)所示。

$$G(v,\omega) = \alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega) \tag{9}$$

式中，$G(v, \omega)$ 为模拟轨迹评价值；$heading(v, \omega)$ 为模拟轨迹末端方向与终点的方位角偏差；$vel(v, \omega)$ 为模拟轨迹速度大小；α、β、γ 分别为指向权值、安全距离权值和速度权值。此外，为了避免某项评价函数占比过大，造成轨迹不平滑，需要对 $heading(v,\omega)$、$dist(v,\omega)$ 和 $vel(v,\omega)$ 进行归一化处理。

$$normal\_Func(i) = \frac{Func(i)}{\sum_{i=1}^{n} Func(i)} \tag{10}$$

2.2　改进 DWA 算法

传统的 DWA 算法在应用过程中存在以下问题：

(1)缺少全局路径的信息。当面对复杂环境时，受评价函数权值的影响，一是导致轮式机器人绕远路；二是容易陷入局部最优解的状况，无法遵循全局最优路径。

(2)机器人在行进过程中，机器人的速度权值处于定值，无法随着障碍物的密度自动调整，一定程度上，降低了效率。

2.2.1　关键点评价子函数 $key(v, \omega)$

全局路径规划的路径，是在静态环境中的最优路径。若轮式机器人在动态避障过程中，能有全局路径作为“参考”，不仅可以避免绕远路，还可以避免陷入局部最优解，遵循全局最优路径。因此，本文引入双向跳点 A* 算法优化后的跳点，作为局部路径规划的关键点，将公式(9)中的终点方向角偏差修改为关键点方向偏差。

$$G(v,\omega) = \alpha \cdot key(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega) \tag{11}$$

2.2.2　自适应速度权值

分析 DWA 算法的原理可知，影响路径的因素包括轮式机器人的线速度和角速度。在传统 DWA 算法中，速度权值取定值，无法根据实际环境发生改变。因此，本文提出自适应速度权值，将影响速度权值的因素分为两类：一类是一定区域内障碍物的数量，一类是该区域障碍物的分布位置。

如图 6 所示，黑色圆形表示轮式机器人，箭头方向为机器人行进方向。将轮式机器人行进区域划分为两个区域，红色区域表示高风险区域，该区域内的

障碍物数量为 c,橙色区域表示低风险区域,该区域内的障碍物数量为 d。则自适应速度权值可由下列公式表示:

$$\gamma = \begin{cases} \gamma_{\max}, & c + d = 0 \\ \gamma_{\max}/(\lambda_1 c + \lambda_2 d + 1/2), & c + d \neq 0 \end{cases} \quad (12)$$

式中,$\gamma_{\max}$ 表示最大速度权值,λ_1 和 λ_2 分别表示高低风险区域权重。其中,$\lambda_1 = 1$,$\lambda_2 = 0.5$。

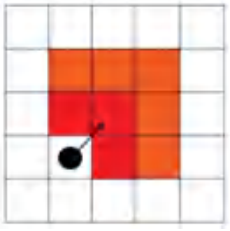


图 6 障碍物权重示意图

Fig. 6 Schematic diagram of obstacle weight

3 算法融合

全局路径规划和局部路径规划往往配合使用,本文使用改进后的双向跳点 A* 算法进行全局路径规划,期望在全局路径规划时减少路径寻优时间、降低遍历节点的数量和缩短路径的长度。改进后的 DWA 算法用于局部路径规划,将双向跳点 A* 算法的跳点作为 DWA 算法的关键点,作为全局路径的关键信息指引机器人前进,并引入自适应速度权值,根据障碍物密度,自动调整速度,兼顾速度和安全性。

综上所述,融合算法的实现步骤如下:

(1)根据传感器获取外部环境来构建地图,并将地图栅格化;

(2)使用双向跳点 A* 算法,找出全局最优路径,并获取路径上的跳点并优化,然后存储到集合 U 中;

(3)将跳点作为 DWA 算法的局部关键点,初始化 DWA 算法的参数;

(4)根据轮式机器人的自身机械特性和环境因素不断进行速度采样,形成动态窗口;

(5)根据评价函数,找出所有采样速度中的最优速度,作为下一时刻的速度;

(6)判断是否达到局部关键点,若未达到,返回步骤(4);若达到局部关键点,选取下一个跳点当做局部关键点;判断是否达到终点,若没有达到,返回步骤(4),若到达则表示路径找到。

4 仿真验证及结果分析

4.1 基本参数

为了验证融合改进双向 A* 算法和 DWA 算法的有效性,本文仿真环境选用 MATLAB2017a,并建立多组对比实验;每个网格代表的实际长度为 1m。其仿真实验具体参数见表 1-3。

表 1 双向跳点 A* 算法参数

Tab. 1 Bidirectional jump point A* algorithm parameters

参数	数值
跳点权重系数 a	3.0
剩余代价权重系数 b	1.0
屏幕刷新时间 t/s	0.1
安全距离 D/m	0.5

表 2 轮式机器人机械特性

Tab. 2 Mechanical characteristics of wheeled robots

参数	数值
最小线速度 $v_{\min}$/(m/s)	0.0
最大线速度 $v_{\max}$/(m/s)	1.0
线加速度 a/(m/s^2)	0.2
最小角速度 $\omega_{\min}$/(°/s)	-20
最大角速度 $\omega_{\max}$/(°/s)	20
角加速度 α/(°/s^2)	50

表 3 DWA 算法参数

Tab. 3 DWA algorithm parameters

参数	数值
障碍物冲突半径判定 R_1/m	0.5
局部目标点半径判定 R_2/m	1.5
时间分辨率 dt/s	0.1
线速度分辨率 dv/(m/s^{-1})	0.01
角速度分辨率 $d\omega$/(°/s)	1
向前预测时间 t/s	3
指向权值 α	0.08
安全距离权值 β	0.2
速度权值 γ	0.1
最大速度权值 $\gamma_{\max}$	0.3

4.2 仿真结果与分析

全局路径规划对比实验如图 7-10 所示。

图 7 传统 A* 算法寻路过程

Fig. 7 The path-finding process of traditional A* algorithm

图 8 双向 A* 算法寻路过程

Fig. 8 The path-finding process of Bidirectional A* algorithm

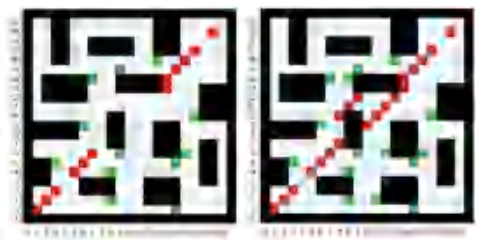


图 9　双向跳点 A* 算法寻路过程

Fig. 9　The path-finding process of Bidirectional jump point A* algorithm

图 10　双向跳点 A* 算法优化

Fig. 10　Optimization of bidirectional jump point A* algorithm

关于传统 A* 算法、双向 A* 算法、双向跳点 A* 算法以及双向跳点 A* 优化算法在寻路时间、路径长度以及拓展节点数量对比见表 4。

表 4　传统 A* 与改进 A* 算法对比

Tab. 4　Comparison between traditional A* and improved A* algorithms

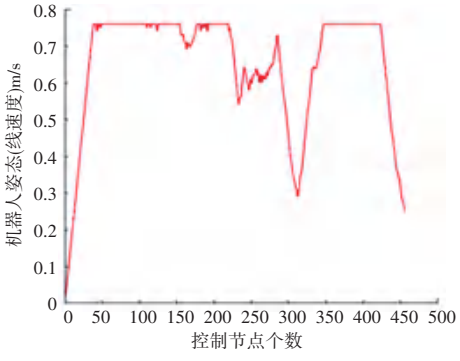
	耗时/s	路径长度/m	拓展节点数量/个
传统 A*	6.538	29.213	87
双向 A*	3.837	30.870	100
双向跳点 A*	3.212	29.213	35
双向跳点 A* 优化	3.212	28.697	35

由表 4 可知：通过融合双向 A* 和 JPS 算法，以及改进算法的评价函数，与传统 A* 算法相比，使得路径规划时间缩短 50.87%，路径长度降低 1.76%，拓展节点数量减少 59.77%。综合来看，提升明显。由图 10 可看出，经过路径平滑优化之后，路径转折次数有所降低，路径更加平滑，且引入的安全距离 D，增加了机器人的安全性。

传统 DWA 算法的安全距离权值 β 均大于指向权值 α 和速度权值 γ，相当于安全优先，但在引入关键点和自适应速度权值后，在相邻关键点之间并无障碍物。因此，对权值做出调整，α 调整为 0.15，β 调整为 0.1，融合改进算法对比实验如图 11-12 所示。

为了对比融合改进算法前后，在表 5 中统计了 DWA、融合改进 DWA 算法的运行时间、路径长度和平均速度。

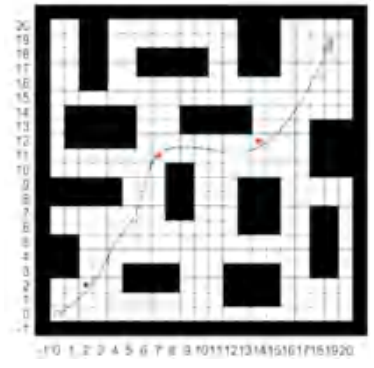
（a）路径轨迹



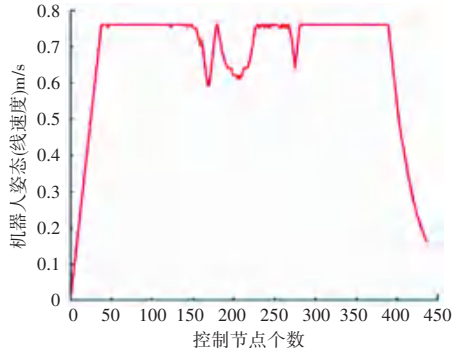
（b）速度曲线

图 11　DWA 算法状态图

Fig. 11　State chart of DWA algorithm



（a）路径轨迹



（b）速度曲线

图 12　融合改进 DWA 算法状态图

Fig. 12　State chart of fusion improved DWA algorithm

表 5 DWA 和融合改进 DWA 算法对比

Tab. 5 Comparison between DWA and fusion improved DWA algorithms

	运行时间/s	路径长度/m	平均速度 m/s
DWA	209.059	29.843	0.623
融合改进 DWA	181.574	28.723	0.694

由表 5 数据可得,融合改进 DWA 算法,使得运行时间缩短 13.14%,路径长度降低 3.75%,平均速度提升 11.40%。分析图 11、图 12 的速度曲线可知,在融入关键点信息以及加入自适应速度权值之后,以指向优先,且速度权值随着障碍物数量和分布自动调整,机器人在障碍物稀疏区域高速前进,在密集区域适当减速,兼顾速度和安全性。

5 结束语

针对轮式机器人路径规划问题,本文提出融合改进 A* 和 DWA 算法。采用双向 A* 和跳点算法相结合来提升全局路径规划效率。仿真实验表明:与传统 A* 算法相比,在耗时、路径长度和拓展节点数量均有所提升。将优化后的全局路径规划的跳点作为 DWA 算法的关键点,使得路径遵循全局最优,而引入的自适应速度权值,根据障碍物数量和分布自动调整速度,使得平均速度提升,提升了效率。

参考文献

[1] 于洋,杜学历,冯迎宾. 融合 A* 算法和动态窗口法的全局动态路径规划方法[J]. 沈阳理工大学学报,2020,39(5):1-7.

[2] FU B, CHEN L, ZHOU Y, et al. An improved A* algorithm for the industrial robot path planning with high success rate and short length [J]. Robotics and Autonomous Systems, 2018, 106: 26-37.

[3] CHAO N, LIU Y, XIA H, et al. Grid-based RRT * for minimum dose walking path-planning in complex radioactive environments [J]. Annals of Nuclear Energy, 2018, 115: 73-82.

[4] LUO Q, WANG H, ZHENG Y, et al. Research on path planning of mobile robot based on improved ant colony algorithm [J]. Neural Computing and Applications, 2020, 32(6): 1555-1566.

[5] OROZCO-ROSAS U, MONTIEL O, SEPÚLVEDA R. Mobile robot path planning using membrane evolutionary artificial potential field[J]. Applied Soft Computing, 2019, 77: 236-251.

[6] FOX D, BURGARD W, THRUN S. The dynamic window approach to collision avoidance[J]. IEEE Robotics & Automation Magazine, 1997, 4(1): 23-33.

[7] 陈娇,徐菱,陈佳,刘卿. 改进 A* 和动态窗口法的移动机器人路径规划[J/OL]. 计算机集成制造系统:1-17[2021-10-12]. http://kns.cnki.net/kcms/detail/11.5946.tp.20201026.1053.026.html.

[8] STENTZ A. Optimal and efficient path planning for partially-known environments[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2002:3310-3317.

[9] KOENIG S, LIKHACHEV M. Fast replanning for navigation in un-known terrain[J].IEEE Transactions on Robotics, 2005,21(3): 354-363.

[10] 杨瑶,付克昌,蒋涛,等. 改进 A* 算法的智能车路径规划研究[J]. 计算机测量与控制,2020,28(10):170-176.

[11] 吴鹏,桑成军,陆忠华,等. 基于改进 A* 算法的移动机器人路径规划研究[J]. 计算机工程与应用,2019,55(21):227-233.

[12] 孔继利,张鹏坤,刘晓平. 双向搜索机制的改进 A* 算法研究[J]. 计算机工程与应用,2021,57(8):231-237.

[13] 卞永明,季鹏成,周怡,等. 基于改进型 DWA 的移动机器人避障路径规划[J]. 中国工程机械学报,2021,19(1):44-49.

[14] 邱磊,刘辉玲,雷建龙. 跳点搜索算法的原理解释及性能分析[J]. 新疆大学学报:自然科学版,2016,33(1):80-87.

[15] 王中玉,曾国辉,黄勃. 基于改进双向 A* 的移动机器人路径规划算法[J]. 传感器与微系统,2020,39(11):141-143,147.

(上接第 13 页)

用流算法最差。最后对生成的 DEM 进行精度验证(由表 3 可知),误差范围为 0~1.4 m,精度较高,中误差为 0.79 m,达到一般地形建模应用要求。

InSAR 可以大面积、快速、高精度、全天时、全天候地获取矿区地表 DEM,对矿区资源利用、控制环境破坏和确保采矿区域的健康绿色发展有着重要意义,可以进行大范围推广。目前 InSAR 获取的 DEM 精度还可以再提高,为矿区发展作出更多贡献。

参考文献

[1] 尹宏杰. 基于 InSAR 的矿区地表形变监测研究[D]. 长沙:中南大学, 2009.

[2] 刘国祥. InSAR 基本原理[J]. 测绘, 2004(4):43-46.

[3] 焦明连, 蒋廷臣. 基于 InSAR 技术矿区地表形变的监测[J]. 淮海工学院学报(自然科学版), 2008(2):75-78.

[4] 靳国旺, 徐青, 杜丽敏. InSAR 干涉图的相位解模糊处理[J]// 仪器仪表学报,2004(S1): 657-659.

[5] 张俊娜, 邓喀中. 基于 GAMMA 软件的 InSAR 数据相位解缠分析比较[J]. 测绘与空间地理信息, 2012(6):171-173.

[6] 罗华, 雷斌, 胡玉新. 一种机载 InSAR 水体阴影的提取和识别方法[J]. 遥感技术与应用, 2014(2):258-263.

[7] 王凯, 方昊然, 杨松林,等. 基于零空间矩阵相位解缠的复杂山区高速公路沿线形变 InSAR 监测[J]. 地理与地理信息科学, 2020, 36(2):32-37.

[8] 王志斌. 星载多通道 SAR/InSAR 成像处理技术研究[D]. 西安:西安电子科技大学, 2017.

[9] 王志勇, 王世超, 孙懿,等. 一种基于 GVF-Snake 模型边界探测的相位解缠算法[J]. 中国矿业大学学报, 2017, 46(6):1394-1401.

[10] 刘子龙, 董臻, 蔡斌,等. 星载 InSAR 非相干图像对精确配准[J]. 遥感技术与应用, 2008(4):94-98.

[11] 葛大庆, 王艳, 范景辉,等. 地表形变 D-InSAR 监测方法及关键问题分析[J]. 国土资源遥感, 2007 (4):14-22.

[12] 祁晓明. PS-InSAR 技术在西安地区的变形监测研究[D]. 西安:长安大学, 2009.